
A Low Energy Realizable Model for Linear Phase Filtering

- A block processing technique -

Master Thesis
Troels Bastholm

Aalborg University
Department of Electronic Systems
Fredrik Bajers Vej 7B
DK-9220 Aalborg



AALBORG UNIVERSITY

STUDENT REPORT

Department of Electronic Systems

Fredrik Bajers Vej 7

DK-9220 Aalborg Ø

<http://es.aau.dk>

Title:

A Low Energy Realizable Model for Linear Phase Filtering

Theme:

Signal Processing

Project Period:

Fall Semester 2014

Project Group:

Group 1078

Participant(s):

Troels Bastholm

Supervisor(s):

Peter Koch

Copies: 2**Page Numbers:** 79**Date of Completion:**

April 13, 2015

Abstract:

This master thesis covers an analysis and design of a forward-backward filtering model to achieve linear phase. The model is realized by the use of a block processing technique. An analytical derivation of the model is carried out and described in details. Further the model is analyzed to conclude how the exact amplitude response of the model should be found. The model is evaluated against a Folded FIR filter which is seen as a reference filter. It is further found that the more narrow the transition band is bigger the advantages to the forward-backward filtering model is in terms of mathematical operations. Fixed-point filters for both methods were found such both filters comply with the same amplitude specification. RTL designs of the two models were developed in order to be able to estimate the energy consumption. It was found that the energy consumption could be lowered by 25 % with the use of the forward-backward filtering model.

The content of this report is freely available, but publication (with reference) may only be pursued due to agreement with the author.

Contents

Preface	ix
1 Introduction	1
1.1 Digital Filter Design Basics	2
1.2 Phase Response	4
1.2.1 Zero phase	4
1.2.2 Linear Phase	4
1.2.3 Non-linear Phase	6
1.3 Non-causal Forward-backward Filtering	6
1.4 Problem statement	7
1.4.1 Hardware Basis	8
1.5 Thesis Outline	9
2 Filter Structures	11
2.1 Realizable Model of Forward-Backward Filtering	11
2.1.1 IIR truncator	11
2.1.2 Forward-Backward filtering model	14
2.2 Structure of Reference filter	20
3 Model Analysis	23
3.1 Mathematical Operations per Sample	23
3.1.1 Forward-Backward filtering model	23
3.1.2 Reference filter	24
3.1.3 Comparison based on filter order	24
3.2 Model Simulation	26
3.2.1 Floating-Point Simulation	27
3.2.2 Numerical Aspects	28
3.3 Zero Input Limit Cycle Oscillation	32
3.4 Amplitude Response Discussion	34
3.5 Phase Response Discussion	37
3.6 Group Delay Considerations	38

4	Fixed-Point Filter Design and Simulation	41
4.1	Design Templates	41
4.2	Filter Design for the Forward-Backward Filtering Model	42
4.2.1	Fixed-Point Simulation of Forward Backward Filtering Model .	43
4.3	Design of Reference Filter	45
4.3.1	Fixed Point Simulation of The Reference filter	45
5	System Design	47
5.1	Design Constraints and Abstractions Levels	47
5.2	Cost Function	48
5.2.1	Supply voltage	49
5.3	Processor Synthesis	50
5.3.1	Forward-Backward Filtering Model	51
5.3.2	Reference Filter	57
5.4	Energy Consumption Estimation	60
6	Conclusion	63
6.1	Future work	64
	Bibliography	65
A	Complexity Test	67
B	Simulation of Model in Matlab	69
B.1	Floating-Point Simulation	69
B.2	Fixed-Point Simulation	70
C	Design of Filters using Matlab	71
C.1	Fixed-Point Filter for the FBF Model	71
C.2	Reference Filter	73
D	Impulse Response and Block Length Considerations	75
E	Implementation of bidirectional Shift Register in VHDL	77
F	Logic Elements	79


List of Acronyms

ADC	Analog to Digital Converter
ASAP	As-Soon-As-Possible
ASIC	Application Specific Integrated Circuit
DFG	Data Flow Graph
DSP	Digital Signal Processing
DTFT	Discrete Time Fourier Transform
DFT	Discrete Fourier Transform
FBF	Forward-Backward Filtering
FIR	Finite Impulse Response
FPGA	Field Programmable Gate Array
FSMD	Final State Machine with Data path
FU	Functional Unit
IIR	Infinite Impulse Response
LIFO	Last In First Out
LE	Logic Elements
LTl	Linear Time-Invariant
PG	Precedence Graph
RAM	Random-Access Memory
RTL	Register Transfer Level
SFG	Signal-Flow Graph
SNR	Signal to Noise Ratio
VHDL	VHSIC Hardware Description Language

Preface

This is the master thesis of Troels Bastholm, Signal Processing and Computing group 14GR1078 at Aalborg University.

Throughout the report chapters, sections, figures, tables are referenced by type and then the number, e.g. figure 3 in chapter 2 is referenced “figure 2.3”. Equations are referenced likewise with a parenthesis, e.g. equation 3 chapter 2 is referenced “equation 2.3”. Citations are in IEEE style, e.g. [7, p. 102] refers to source number 7 in the reference list page 102.

All software developed in this project along with an electronic copy of the report is available on the attached appendix-CD. The developed simulations are implemented in Matlab, and require the fixed-point designer package. The VHDL code can be synthesized in the Altera Quartus II design suite. A referenced to code on the appendix-CD is done with stating the path on the CD as e.g. this reference to the matlab folder ./Matlab.

The author would like to give a special thanks to Peter Koch for supervision and inputs during the project.

Aalborg University, April 13, 2015

Troels Bastholm
<tbasth09@student.aau.dk>

Chapter 1

Introduction

Digital Signal Processing (DSP) is today applied in a wide range of applications within science and engineering. These applications are more and more often battery powered devices making energy consumption is a critical parameter. In many of these applications, linear phase filters are used. These filters are designed as Finite Impulse Response (FIR) filters, which consumes more than Infinite Impulse Response (IIR) filters. The reason to use a FIR filter is because a causal IIR filter is unable to have a linear phase [8, p. 236]. This is a very complex problem and recent studies have shown how an IIR filter can be designed to have approximately linear phase by using different optimization methods, [7, 6, 12]. Each method implies different filter constraints to the amplitude response and they all share that they only approximate a linear phase.

In this project a real time implementation with the use of IIR filters with no constraints to the frequency response is designed and can replace any linear phase FIR filter. To achieve linear phase with any IIR filter, a filtering method called forward-backward filtering can be applied. The method is also known as bidirectional filtering or zero phase filtering, but in this project always mentioned as Forward-Backward Filtering (FBF). In [11] a method suggests how to implement FBF with the use of IIR filters by using block processing, which will be mentioned as the FBF model. The project will evaluate the FBF model against a FIR filter as a reference. Both filter's fixed point implementations have to comply with the same amplitude specification. Following objectives define the scope of this project.

Objectives:

1. Analyzing the FBF model proposed by [11].
2. Design a realizable fixed point systems of the FBF model, and the reference filter which is a FIR filter.
3. Compare FBF model by the use of IIR filters with the reference filter to determine in which cases the FBF model consumes less energy when implemented on a fixed-point platform, and how much energy that can be saved.

This project does not consider any specific application, but is a study on how to implement a low power linear phase filter. For the rest of this chapter the parameters in filter design will be clarified and the relevant theory of linear phase filtering is described. Then, in section 1.3 FBF is discussed in theory and in terms of its frequency response.

1.1 Digital Filter Design Basics

The following section gives a brief introduction of digital filter design and the parameters. A digital filter extracts the relevant information of a signal. It is done by feeding the input $x[n]$ sample by sample to a filter, generating the output $y[n]$. The filter has modified the information of the input according to the filter specifications which also results in a phase shift. The core of this project is to design a linear phase filter which uses as little energy as possible. Since no application is connected to the project, a case-study of an low-pass filter is considered. When designing a low-pass filter, the cut-off frequency is used to construct the ideal frequency response, $D(\omega)$. This frequency response is approximated by the transfer function $H(\omega)$ with is found for an IIR or FIR filter. A digital filter is always periodic for every 2π radians of the variable ω , and the frequency region is usually defined from $[-\pi, \pi]$. A filter with only real coefficients has a frequency response, which is conjugate-symmetric meaning that $D^*(\omega) = D(-\omega)$. If a filter have conjugate-symmetric frequency response, it is sufficient to define the positive frequencies. In a simple case study an ideal low-pass filter with zero phase has the following frequency response:

$$D(\omega) = \begin{cases} 1, & |\omega| < \omega_c \\ 0, & \omega_c < |\omega| < \pi \end{cases} \quad (1.1)$$

where ω_c is the cut-off frequency, and is corresponding to the transition between one and zero in the ideal amplitude response of the low-pass filter. The ideal frequency response from equation (1.1) is plotted in figure 1.1, and it is clear to see the sharp

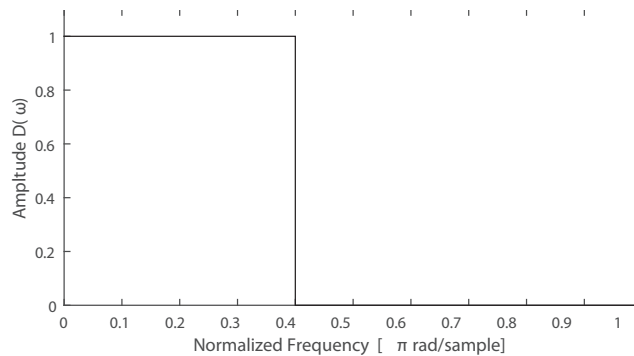


Figure 1.1: Ideal frequency response for a low-pass filter $D(\omega)$.

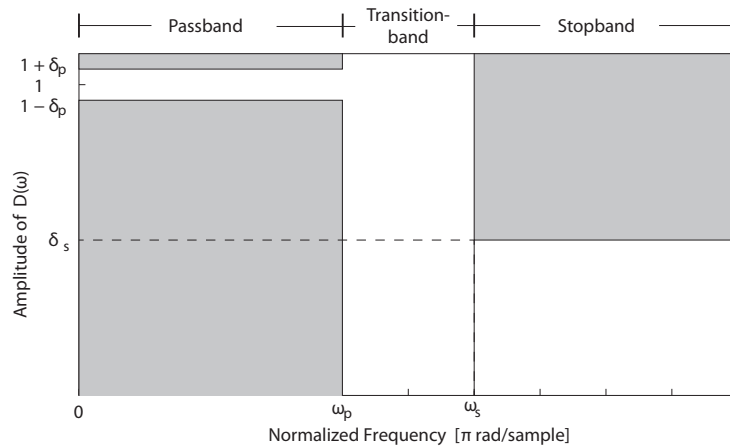


Figure 1.2: The figure shows the design template with parameters for a one sided amplitude response of a low-pass filter.[8, p. 230]

cut-off edge. An approximation of this transfer function in discrete time can be designed using a digital filter. To design this approximation different design methods are available. These design methods use some constraints to fit the amplitude response of the filters transfer function. These design constraints can be seen in figure 1.2. The design constraints for the three bands in a low-pass filter are defined as following:

- The passband has a bandwidth ω_p . This is the frequency range where the needed information is located. Maximum passband ripple δ_p defines how much the filter can diverge from the desired gain in the passband region.
- The transition has a bandwidth of $\omega_s - \omega_p$. The ideal cut-off characteristic is unrealizable but the transition band specifies where the attenuation should be changed between passband ripple and stopband attenuation. Cut-off frequency ω_c is the -3 dB point in amplitude response and this point is normally between ω_p and ω_s depending on the size of the ripple in the filter.
- The stopband has a bandwidth of $\pi - \omega_s$. This is the frequency range where no information is neither wanted nor needed. Stopband attenuation δ_s defines the maximum stopband attenuation.

These constraints are used and have to be taken into account when a filter is designed. A filter which is suitable for the design template in figure 1.2 has a transfer function $H(\omega)$. The next objective is to fit either a FIR or an IIR filter design into the template. Normally the design method within the chosen filter type with lowest order filter is desired in order to have the lowest number of mathematical operations. The description in this sections is based on the material in [8].

These design methods only consider the specification of the amplitude response and have no constraints on the phase response. This is the traditional method to

design a filter which complies to the required specifications, but if some constraints also are made on the phase, these should be considered in the design phase as well. There are three types of phase response. Linear phase, non-linear phase and zero phase. The following subsections will describe these characteristics in further details.

1.2 Phase Response

As previous mentioned, a filter can have three types of phase responses. One example of each of these types are listed in figure 1.3, including impulse response, phase response and pulse response. A phase response determines the phase shift at frequency ω from the input to the output of a transfer function. The phase response is defined as the argument of the frequency response.

$$\theta(\omega) = \arg H(e^{j\omega}) = \arctan \left(\frac{\Im(H(e^{j\omega}))}{\Re(H(e^{j\omega}))} \right), \quad |\omega| < \pi \quad (1.2)$$

The phase shift in radians is given by $\theta(\omega)$ in equation (1.2), where $H(e^{j\omega})$ is the Frequency response. The derivative of $\theta(\omega)$ is the slope of the function and will describe the group delay of the transfer function. [10, p.257]

$$\tau_{gd}(\omega) = -\frac{d\theta(\omega)}{d\omega} \quad (1.3)$$

These functions will have different characteristics depending on the transfer function and will be described further in the following.

1.2.1 Zero phase

As the name implies a filter with zero phase introduces zero shift, which also gives it a constant group delay of zero. As seen in figure 1.3a it requires a symmetric impulse response around sample of index number zero. A filter with this kind of impulse response is impossible to implement in a real-time system since it is non-causal. A non-causal system's output depends on future input and can therefore only be applied with post processing filters. If a zero phase system is applied as post processing the phase response will be as in figure 1.3b. The pulse response in figure 1.3c display what happens when a signal takes a positive step followed after 50 samples by a negative step. It should be noted that for a zero phase filter with a symmetric impulse response, the rising edge is equal to the time reversed falling edge.

1.2.2 Linear Phase

A filter is said to have linear phase when all frequency components have the same group delay. This results in a phase response like the one showed in figure 1.3e. A linear phase response is achieved by having a filter with an even or odd symmetric

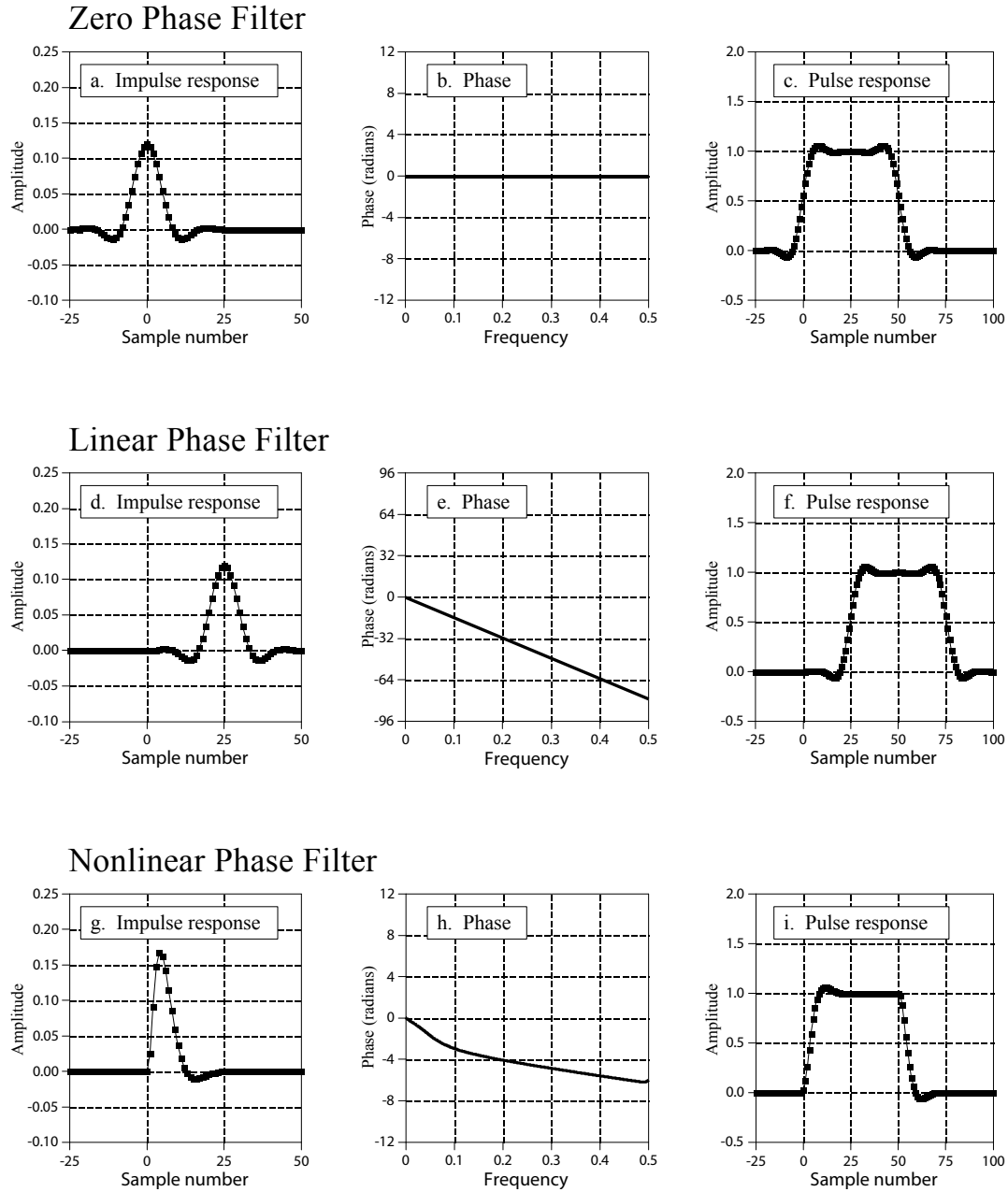


Figure 1.3: Impulse response, phase response and pulse response are shown with an example of each type of filter; zero phase, linear phase and non-linear phase. The impulse response for zero phase has a rectangular windows applied for sample number $[-25; 25]$ and the linear phase impulse response has a rectangular window applied for sample number $[0; 50]$. The non-linear phase impulse response has an identical window applied as the linear, with a non-symmetric around sample index number 25. [14, p. 329].

impulse response, even symmetry is seen in figure 1.3d and is identical to figure 1.3a with the exception that time has been shifted, so all non-zero values are on the positive side of time zero. This method gives the same edges as on the zero phase pulse response, but shifted with the same number of samples as the impulse response. This is shown in figure 1.3f. Comparing linear phase system with zero phase system, it is possible to see that zero phase system is a special case of linear phase system. A filter with these characteristics can be implemented as a FIR filter, if the system is causal as in this example.

1.2.3 Non-linear Phase

Non-linear phase is achieved when the group delay is no longer constant over the frequency. An example of how a non-symmetric impulse response of a non-linear phase system is showed in figure 1.3g. This makes the phase response non-linear as showed in figure 1.3h where the line is no longer straight, and at last the pulse response in figure 1.3i, and it is not symmetric as the previous. Both FIR and IIR filters can result in these characteristics.

The description in this sections is based on the material in [14, p. 328]

1.3 Non-causal Forward-backward Filtering

A method to achieve zero phase is the FBF process. Forward-backward filtering consist of a cascade with two identical filters and time reversals. Since the process consist of time reversal processes, it is non-causal and can only be applied on finite number of samples as post processing. In figure 1.4 a block diagram for the filtering

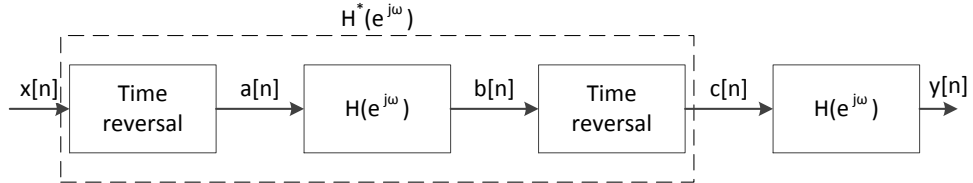


Figure 1.4: Block diagram of the FBF processes, where $a[n]$ and $y[n]$ is equal to $x[-n]$ and $c[-n]$ respectively, and $H(e^{j\omega})$ is identical filters.

process is shown. To get an idea of the performance of this filter, the transfer function is derived. By use of the time reversal property of the Discrete Time Fourier Transform (DTFT) then if $x[n] \xrightarrow{DTFT} X(e^{j\omega})$, then $x[-n] \xrightarrow{DTFT} X^*(e^{j\omega})$. Then we can derive following:

$$A(e^{j\omega}) = X^*(e^{j\omega}) \quad (1.4)$$

$$B(e^{j\omega}) = A(e^{j\omega})H(e^{j\omega}) = X^*(e^{j\omega})H(e^{j\omega}) \quad (1.5)$$

$$C(e^{j\omega}) = B^*(e^{j\omega}) = X(e^{j\omega})H^*(e^{j\omega}) \quad (1.6)$$

$$Y(e^{j\omega}) = C(e^{j\omega})H(e^{j\omega}) = X(e^{j\omega})H^*(e^{j\omega})H(e^{j\omega}) \quad (1.7)$$

$$\frac{Y(e^{j\omega})}{X(e^{j\omega})} = H^*(e^{j\omega})H(e^{j\omega}) \quad (1.8)$$

In equation (1.8) the transfer function for FBF is given which is the effective transfer function $H_{eff}(e^{j\omega})$ of the processes from the input $x[n]$ to the output $y[n]$ seen in figure 1.4. A complex number multiplied by its own complex conjugate always results in a real number, thus

$$H_{eff}(e^{j\omega}) = H^*(e^{j\omega})H(e^{j\omega}) = |H(e^{j\omega})|^2 \quad (1.9)$$

As it can be seen in equation (1.9) the effective frequency response of FBF equals the squared amplitude response of the used filters. The squared amplitude response yields the double gain in dB due to logarithmic calculation rules. In the design process the amplitude response $|H(e^{j\omega})|$ should therefore only have half the gain required of the effective amplitude response $H_{eff}(e^{j\omega})$. The effective frequency response describes input/output relation for FBF. This method can be applied with the use of any Linear Time-Invariant (LTI) filter, and no more restrictions is added to the design method.

1.4 Problem statement

Above the non-causal FBF was defined. This method can be applied with a block processing method as [11] proposes, which is called the FBF model. This model claims to have a lower number of multiplications when performing linear phase filtering than an ordinary FIR filter. The focus during this project will be to analyze the FBF model that is using a block-wise filtering approach. The target of the project will be to see how much energy can be saved by using this method. As mentioned above, the FBF model is evaluated against a reference filter that will be a FIR filter.

In order to create an overview of the project the A^3 model is shown in figure 1.5. The model describes a general approach where the three steps visualize the design process. In the application circle some specifications are given. These specifications can be mapped to any algorithm that comply with the given design specifications. Each of the Algorithms can then be mapped on to a architecture which can execute the algorithm. The three design spaces describe the A^3 model. In figure 1.5 it can be seen that one set of specifications is mapped onto two different algorithms. In this project these two algorithms is the FBF model and the reference filter. These two

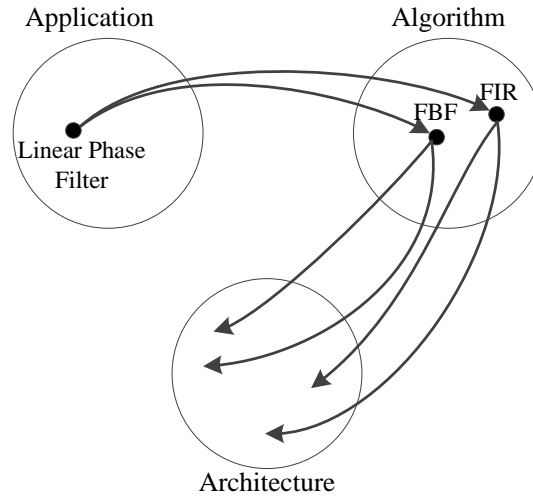


Figure 1.5: The A^3 model describing the mapping from Application to Architecture.

algorithms can be mapped in various ways to a fixed-point platform. It is generally known as a one-to-many mapping but for this project the mapping will be conducted by considering energy consumption in the mapping process. This means that the result will be two implementations that can be evaluated against each other in terms of the energy consumption.

1.4.1 Hardware Basis

The target platform for the two implementations will be an Altera Cyclone II 2C35 FPGA [5]. This platform includes 35 embedded 18-bit multipliers that can be used for testing the implementation. The target would be to map the final result to an Application Specific Integrated Circuit (ASIC) but with the FPGA it is possible to describe the implementation in terms of logic elements. The Field Programmable Gate Array (FPGA) contains 33216 Logic Elements (LE)s or also known as logic cells, but Altera mentions them as LEs. Each LE can be programmed to perform the needed 1 bit operation to achieve the Register Transfer Level (RTL) specification that is designed in the implementation. Since no application is connected to the project, it is chosen to use the constraints from the Analog to Digital Converter (ADC) on the Altera DE2 kit and the 50 MHz oscillator. For the ADC it is chosen to use the 44.1 kHz sample rate which is a typical sample rate for music.

1.5 Thesis Outline

This section aims to give a clear overview of the project structure.

Chapter 2:

Description of the FBF model and an in-depth analytic derivation of the FBF model, including simple examples that clarifies the theory. Afterwards the reference filter is defined.

Chapter 3:

An analysis of the FBF model in terms of number of required mathematical operations. Further a verification of the frequency response is discussed and it is concluded how to determine the block length for the FBF model.

Chapter 4:

To design the reference filter coefficients and the IIR filter coefficients for use in FBF model a design template is found on base of the knowledge gained in chapter 3. The filter coefficients are found and the SNR to the two models are found.

Chapter 5:

In this chapter the model is mapped onto a RTL design by using precedence graphs where the scheduling is specified. Both models are defined such that all needed registers and their executing frequency is known. This information is used to give an estimate of energy consumption of each model which then is compared.

Chapter 2

Filter Structures

Before going into details about the filter design for the real-time implementations of the two filters, some more knowledge on the FBF method is needed. This chapter aims to give an introduction to a realizable model of the FBF method with the use of IIR filters by [11]. This realizable model gives some inherent drawbacks to the frequency response as compared to the post processing method described section 1.3. These side effects will be described in the following.

2.1 Realizable Model of Forward-Backward Filtering

The realizable model of FBF proposed by [11] is a block-wise implementation. The block-wise implementation have a block length L , and each block is filtered by an IIR filter. In theory an IIR filter have, as the name implies, an infinite impulse response. In this case the filter will be implemented on a fixed-point platform resulting in a finite word length for coefficients, state variables, and input/output. This results in a finite impulse response since the impulse response at some point will decay to a value less than the precision chosen for the fixed-point system. When the impulse response decays to a value less than the fixed-point precision, the output will be truncated to zero. Unfortunately this is not always the case since fixed-point IIR filters can suffer from what is known as zero input limit cycles, which is explained further in section 3.3. At this point the length of this impulse response of the IIR filters is said to be finite and defined as length L . To describe a block diagram for this implementation, the system needs to be derived analytically. This is done in the next two sections where an IIR truncator is derived analytically in the first section and the Forward-backward filtering model in the second.

2.1.1 IIR truncator

Before going into the derivation of the FBF model, the overlap-add method needs to be described, since it is a key element for the final system. The overlap-add method is based on filters of finite responses and, based on previous discussion, the method

is used with IIR filters. The method is used to develop an IIR truncator in figure 2.2, which is a basic element in the final model in figure 2.9. The input for the overlap-add method is divided into K blocks and definition of the input signal $x[n]$ in block number k will be defined as follows:

$$x[n] := \sum_{k=0}^{\infty} x_k[n] \quad (2.1)$$

$$x_k[n] := \begin{cases} x[n] & kL \leq n \leq (k+1)L - 1 \\ 0 & \text{otherwise} \end{cases} \quad (2.2)$$

In equation (2.1) and (2.2) $x[n]$ is now defined in block of data. The previous definition can be used to describe the block-wise input $x_k[n]$, as seen in figure 2.1. Since

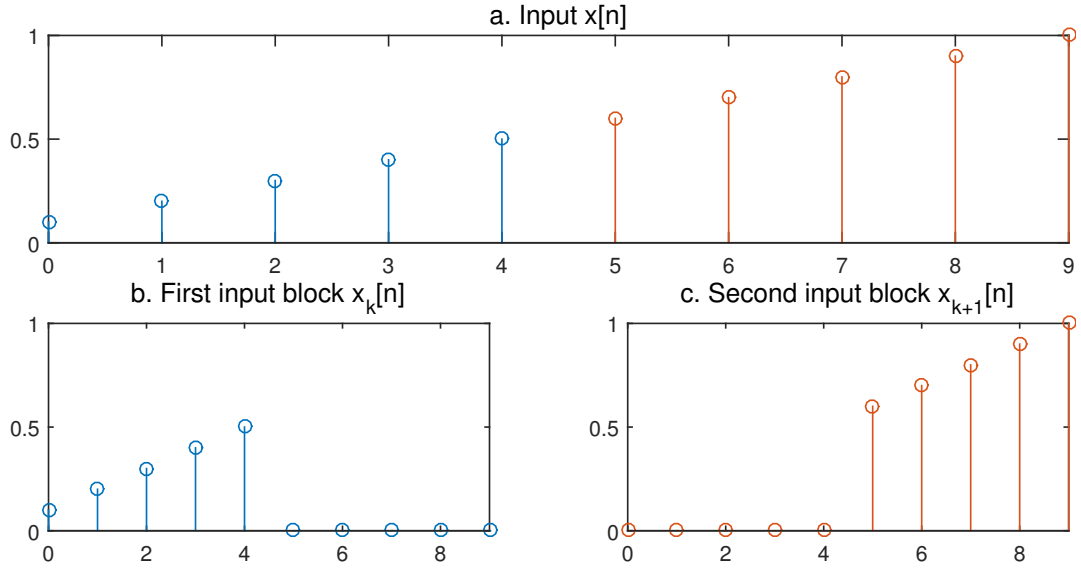


Figure 2.1: Example of using equation (2.2) with a block length of $L=5$ samples. The plots have sample number n on the x-axis and amplitude on the y-axis.

the impulse response have length L , a convolution of input $x_k[n]$ with the impulse response $h[n]$, will result in a corresponding output section of length less than or equal to $2L - 1$. Equation (2.3) defines the standard convolution of a finite impulse response filter. With use of the additivity property of the superposition principle,

equation (2.4) defines the contribution from each of the blocks [10, p. 18].

$$y[n] = \sum_{m=n-L}^n x[m]h[n-m] \quad (2.3)$$

$$y[n] = \sum_{m=n-L+1}^{kL-1} x_{k-1}[m]h[n-m] + \sum_{m=kL}^n x_k[m]h[n-m] \quad (2.4)$$

$$y[n] = y_{k-1,tail}[n] + y_{k,lead}[n] \quad (2.5)$$

$$y_k[n] = y_{k,tail}[n] + y_{k,lead}[n] \quad (2.6)$$

Equation (2.5) describes the same as equation (2.4) just with a shorter notation for further use. Furthermore, equation (2.5) specifies the "Leading response" denoted $y_{k,lead}[n]$ and the "Tailing response" denoted with $y_{k,tail}[n]$. Equation (2.6) describes the output of one single input block filtered in the system, while equation (2.5) describes the output for one block of data fed to the input. A block diagram rep-

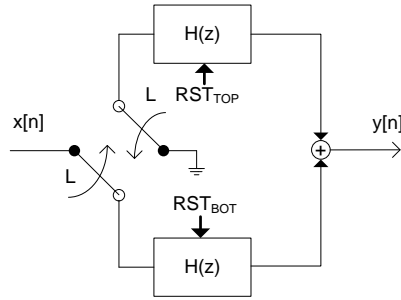


Figure 2.2: Block diagram of overlap-add IIR truncator, where $H(z)$ is identical filters with finite impulse response of length less than or equal to L . The gates will switch for every L samples and shown in BOT position. [11]

resenting equation (2.5) can be seen in figure 2.2. The block diagram shows that the input is split into blocks of length L , which are filtered separately and combined into one output. $H(z)$ are identical filters with an impulse response of length less than or equal to L . If identical IIR filters and infinite word length is used for this structure, the impulse will be infinite. If the impulse response of such IIR filter is

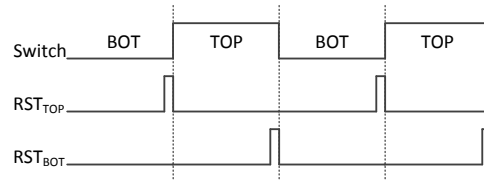


Figure 2.3: Timing diagram for overlap-add IIR truncator in figure 2.2. Switch refers to how $x[n]$ is fed to either the top filter or the bottom filter. [11]

truncated by resetting all internal registers after L samples of zero inputs, the filter will no longer behave as an LTI filter. This is why it is important that the filter do not suffer from zero input limit cycles and has a finite impulse response. The switch change every L samples according to the timing diagram in figure 2.3. A convolution for an input block of length L and an impulse response of length L , equals a output of $2L-1$ samples. This leaves 1 sample period to reset the filter before one new input block is processed in the filter. The timing to reset the filters is shown in figure 2.3. To give an idea of how the block diagram in figure 2.2 is working, an example

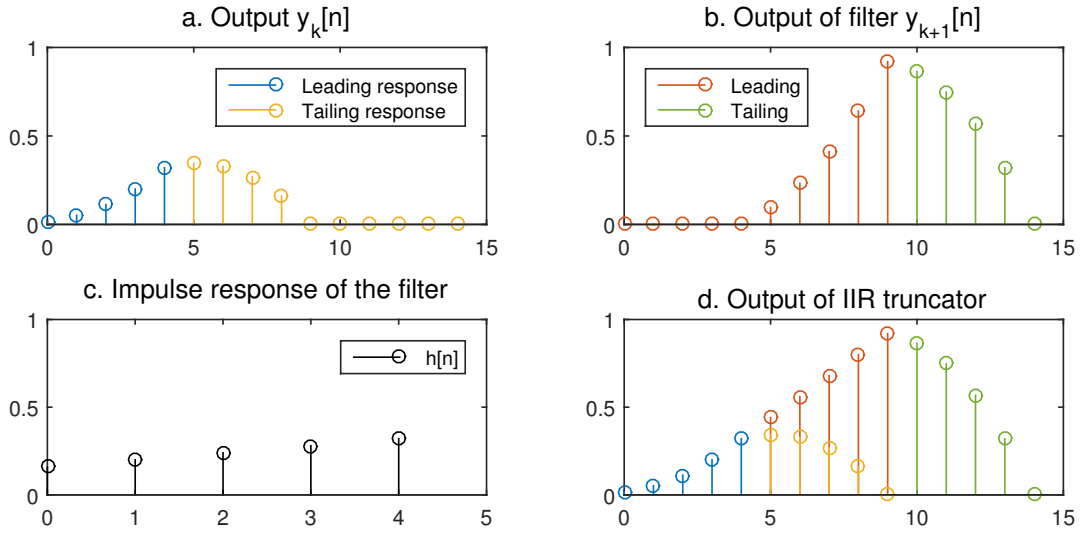


Figure 2.4: Example of using equation (2.5) with a block length of $L=5$ samples. The plots have sample number n on the x-axis and amplitude on the y-axis. The signals in a and b are c convoluted with the signals in figure 2.1, while the signal in d is a and b added together.

in figure 2.4 shows a filter on an input signal from figure 2.1 which has length $2L$. The input blocks from figure 2.1b and 2.1c respectively have been convoluted with the impulse response in figure 2.4c and the output is shown in figure 2.4a and 2.4b respectively. Each signal is shown in two different colors, indicating whether it is the leading response or the trailing response. Each signal is added together and the output can be seen in figure 2.4d.

2.1.2 Forward-Backward filtering model

The overlap-add IIR truncator can also be modified and used to realize the FBF model simply by introducing a time delay. To describe how the final block diagram is developed, an analytically derivation is needed. Each input block $x_k[n]$ needs to be time reversed and defined as $a_k[n]$ later a signal $b_k[n]$ is block-wise time reversed

to $c_k[n]$ as seen in equation (2.7) and (2.7) respectively.

$$n' = (2k + 1)L - 1 - n$$

$$a_k[n] = x_k[n'] \quad (2.7)$$

$$c_k[n] = b_k[n'] \quad (2.8)$$

This definition is a non-causal process, since the last value for a block is moved to the first within the same block. The process will be realized as a Last In First Out (LIFO) buffer such $n' = (2k + 2)L - 1 - n$, which introduces a delay to all samples and make it a causal process. In figure 2.5 the blue line shows the timing of the non-causal

k	0	0	0	0	0	1	1	1	1	1	2	2	2	2	2
n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$x[n]$	$x[1]$	$x[2]$	$x[3]$	$x[4]$	$x[5]$										
$a[n]$	$x[5]$	$x[4]$	$x[3]$	$x[2]$	$x[1]$										
$a[n]$						$x[5]$	$x[4]$	$x[3]$	$x[2]$	$x[1]$					
$c[n]$	$x[1]$	$x[2]$	$x[3]$	$x[5]$	$x[5]$										
$c[n]$											$x[1]$	$x[2]$	$x[3]$	$x[4]$	$x[5]$

Figure 2.5: Timing schedule for $x_k[n]$, $a_k[n]$ and $c_k[n]$ with a block length L equal 5 where the blue timing of $b_k[n]$ is used in the analytic derivation and the green timing $b[n]$ is one which is possible to realize. Same procedure regards $c_k[n]$, and the timing diagram is based on that $b_k[n] = a_k[n]$. The individual columns represent consecutive time to sample number n .

process and the green shows the timing of the causal. In this analytic derivation the causal process, would make the notation of block number k non-transparent later on. In order to keep it as simple as possible, the non-causal process is used for the analytic derivation. In figure 2.5 it can also be seen that the total delay which is ignored equals $2L$ samples.

The non-causal filtering shown in equation (2.9) is what is wanted to realize, but due to the non-causality it is not possible. By block-wise time reversing the input as in equation (2.7), the order of $x[n]$ is no longer contiguous when described by $a[n]$. This leads to what is stated in equation (2.10), where two convolutions are needed to multiply the same indexes as seen in figure 2.6. The trick in equation (2.10) also makes it possible to use forward filtering recursive filters again. A problem in equation (2.10) is the fact that it is still non-causal. In equation (2.11) a $2L$ sample delay has been added, such that all samples needed for the calculation is less than or equal to sample number n .

$$c[n] = x[n] * h[-n] \quad (2.9)$$

$$b[n] = a_k[n] * h[n] + a_{k+1}[n + 2L] * h[n] \quad (2.10)$$

$$b[n - 2L] \Rightarrow b[n] = a_{k-2}[n - 2L] * h[n] + a_{k-1}[n] * h[n] \quad (2.11)$$

$$b[n] = b_{k-2,lead}[n - 2L] + b_{k-1,tail}[n] \quad (2.12)$$

Figure 2.6 shows an example of how one output is computed. The non-causal convolution in equation (2.9) is shown in figure 2.6a. This shows which indexes that

are needed to be multiplied and added together for $n=3$. The leading convolution ($a_{k-2,lead}[n-2L]*h[n]$) in equation (2.11) is shown in figure 2.6b, which also is known as the tailing response. The tailing convolution ($a_{k-1,tail}[n]*h[n]$) is shown in figure 2.6c. The result of adding the leading response together with the tailing response

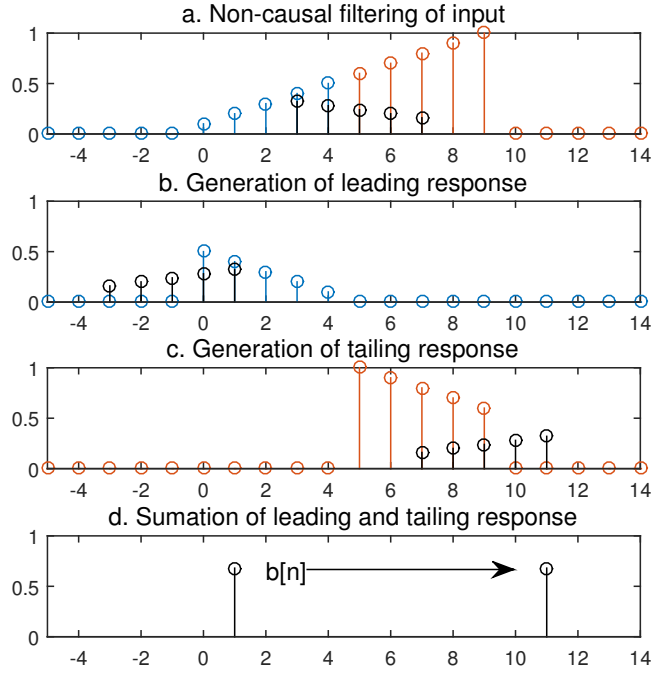


Figure 2.6: This is an example based on the same input as in figure 2.1 and the used filter from figure 2.4c. The non-causal filtering input in figure a is an example of filtering $x[n] * h[-n]$, and in this example $n=3$. In figure b and c is an example of equation (2.9) and the block number $k=2$, $L=5$ and $n=11$. Figure c pictures the $2L$ sample delay that was introduced in equation (2.11), to make the function causal.

is shown in figure 2.6d where it also is possible to see how the delay introduced in equation (2.11) moves the result. On the base of the convolution for the overlap-add IIR truncator in equation (2.4), a similar equation to (2.11) for the time reversed and filtered sequence ($b[n]$) is derived.

$$b[n] = \sum_{m=n-L+1}^{kL-1} a_{k-1}[m]h[n-m] + \sum_{m=kL}^n a_{k-2}[m-2L]h[n-m] \quad (2.13)$$

$$b[n] = b_{k-1,tail}[n] + b_{k-2,lead}[n-2L] \quad (2.14)$$

$$b_k[n] = b_{k,lead}[n] + b_{k,tail}[n] \quad (2.15)$$

In equation (2.14) the leading response and the tailing response is specified in the same manner as previous. Studying equation (2.13), it is possible to see that when $n=kL$ the result of the tailing response $b_{k-1,tail}[n+2L]$ will always equal zero. This

is the sample period where the filter is resetting all values to zero as seen in the overlap-add IIR truncator. In the example in figure 2.7 it is clear to see that the last tailing response equals zero. Figure 2.7a and 2.7b shows the result of a signal

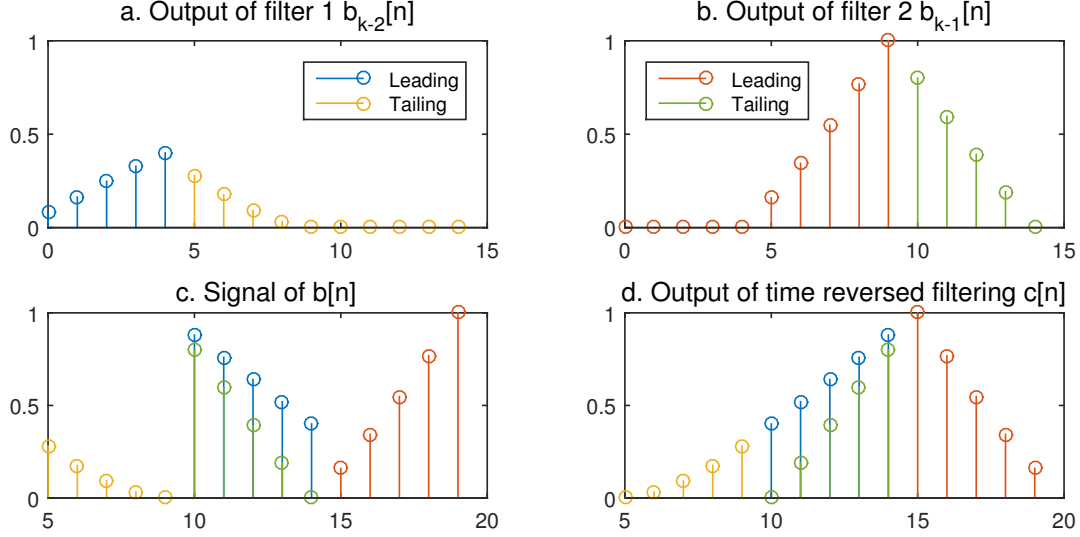


Figure 2.7: The result of the convolution in figure 2.6b and 2.6c is plotted in figure a and b. In figure c these are added together where the colors from each contribution are kept, such the leading and tailing responses can be traced back to figure a and b. In figure d the block-wise time reversal from equation (2.8) is applied on the signal $b[n]$.

length of $2L$ samples being convoluted with the two identical filters and combined according to equation (2.14) in figure 2.7c. In 2.7d the signal is block-wise time reversed and identical to what was seen for the Overlap-add IIR truncator in figure 2.4d. For reader convince equation (2.11) is restated in equation (2.16) to derive the time reversed signal of $b[n]$.

$$b[n] = a_{k-2}[n - 2L] * h[n] + a_{k-1}[n] * h[n] \quad (2.16)$$

$$b[n] = x_{k-2}[n' - 2L] * h[n] + x_{k-1}[n'] * h[n] \quad (2.17)$$

$$c[n] = x_{k-2}[n - 2L] * h[-n] + x_{k-1}[n - 2L] * h[-n] \quad (2.18)$$

$$c[n] = x[n - 2L] * h[-n] \quad (2.19)$$

The result in equation (2.19) shows that the reverse filtering of $x[n]$ simply delayed by $2L$ samples is achieved. An assumption to the derivation was that the block-wise time reversal is processed with no delay. In figure 2.5 it is concluded that these two block-wise time reversals of length L contributes together with a $2L$ sample delay. This delay and the $2L$ sample delay that was seen in equation (2.19) gives a total of $4L$ sample delay. The frequency response for the ideal delay $h_{id}[n]$ where $n_d = 4L$,

as it is described in [10, p. 204]

$$h_{id}[n] = \delta[n - n_d] \quad (2.20)$$

$$H_{id}(e^{j\omega}) = e^{-j\omega n_d} \Rightarrow e^{-j\omega 4L} \quad (2.21)$$

$$\theta_{id}(e^{j\omega}) = -\omega n_d \Rightarrow -\omega 4L, \quad |\omega| \leq \pi \quad (2.22)$$

Adding this ideal delay to the complex conjugate filter, the model will introduce the following frequency response for the system.

$$H^*(e^{j\omega})e^{-j\omega 4L} \quad (2.23)$$

In figure 2.8 a block diagram of the analytic derivation is shown. This block diagram

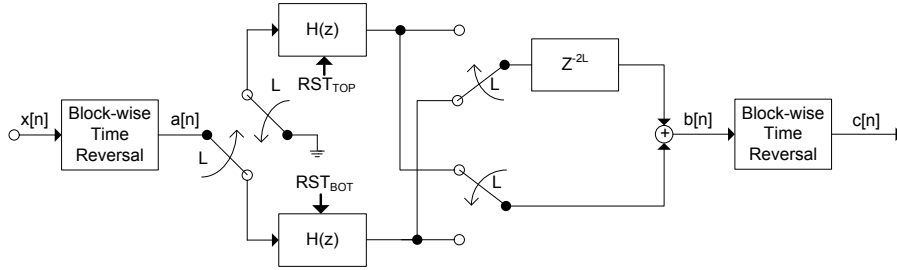


Figure 2.8: A block diagram for a block-wise implementation of reverse filtering of the filter $H(e^{j\omega})$, where the length of the impulse response is less than or equal to L . The block-wise time reversal is of length L corresponds to LIFO buffer with L samples. $H(z)$ are identical filters which are reset according to the timing diagram in figure 2.3.

follows the same timing as the IIR truncator shown in figure 2.3. This block diagram can be put in cascade with a standard forward filtering IIR filter with the same specification as the one in the reverse filtering part, as seen in figure 2.9. This adds a convolution to equation (2.19) as seen in equation (2.24) and the frequency response changes to what is shown in equation (2.25).

$$y[n] = h[n] * c[n] \quad (2.24)$$

$$H_{FBF}(e^{j\omega}) = H(e^{j\omega})H^*(e^{j\omega})e^{-j\omega 4L} \quad (2.25)$$

$$H_{FBF}(e^{j\omega}) = |H(e^{j\omega})|^2 e^{-j\omega 4L} \quad (2.26)$$

$$|H_{FBF}(e^{j\omega})| = |H(e^{j\omega})|^2 \quad (2.27)$$

In equation (2.26) it can be seen that the phase response of the filter $H(e^{j\omega})$ do not comply with any phase response to the overall frequency response $H_{FBF}(e^{j\omega})$, hence this part of the equation is purely real. The phase response only originates from the delay which is expressed as the exponential part $e^{-j\omega 4L}$, which contributes with a linear phase. The group delay of the transfer function will then depend on L as

follows:

$$\theta_{FBF}(\omega) = \arg |H(e^{j\omega})|^2 e^{-j\omega 4L} = \arg e^{-j\omega 4L} = -\omega 4L, \quad |\omega| \leq \pi \quad (2.28)$$

$$\tau_{gd,FBF}(\omega) = -\frac{d\theta_{FBF}(\omega)}{d\omega} = 4L \quad (2.29)$$

As seen above in equation (2.28) the filter will contribute with a constant group delay. The slope of the phase response is decided by the length of L , since the group delay is found to be $\tau_{gd,FBF} = 4L$. Since this is in terms of a normalized filter with a sample rate of 1 Hz this means the normalized filter have $4L$ seconds group delay. The final block diagram can be seen in figure 2.9 and the only difference between

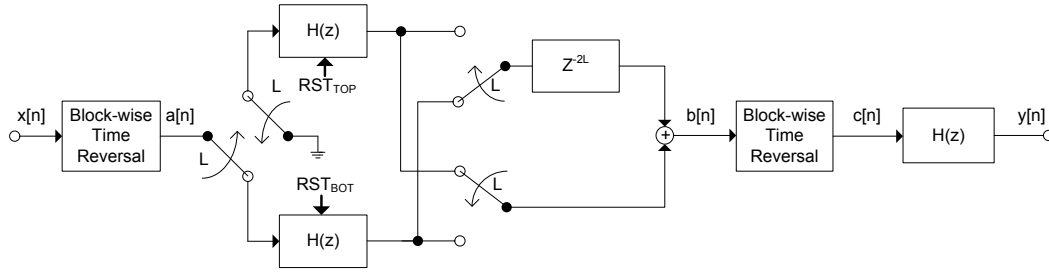


Figure 2.9: Block diagram for block-wise implementation of FBF. The diagram is based on the same assumptions as figure 2.8. [11]

that and figure 2.8 is that a forward filtering has been cascaded. All filters are IIR filters. The algorithm for an IIR filter is given by the following.

$$y[n] = \sum_{i=0}^K b_i x[n-i] - \sum_{i=1}^K a_i y[n-i] \quad (2.30)$$

In equation 2.30 it is assumed that $a_0 = 1$ and that the order of the feed-forward filter is the same as the order of feed-backward filter.

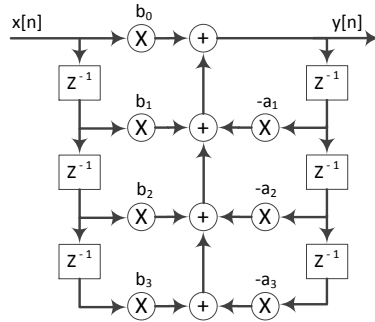


Figure 2.10: Data Flow Graph of a standard IIR filter with order $K=3$.

The filters are going to be implemented on a fixed-point platform where the dynamic range is lower than floating point, and a direct form 1 IIR filter has been chosen. In figure 2.10 a Data Flow Graph (DFG) of a direct form 1 filter of order K can be seen.

2.2 Structure of Reference filter

The reference filter is a linear phase FIR filter with symmetric coefficients. A FIR filter operates as a convolution since the FIR filters coefficient describe its impulse response. The equation for a FIR filter is given as:

$$y[n] = \sum_{k=0}^K h[k]x[n-k] \quad (2.31)$$

where $x[n]$ is the input, $y[n]$ is the filtered output, $h[n]$ is the impulse response of the filter and K corresponds to the order of the filter. In figure 2.11 the equation is converted into a DFG. As mentioned in section 1.2.2 a linear phase FIR filter has

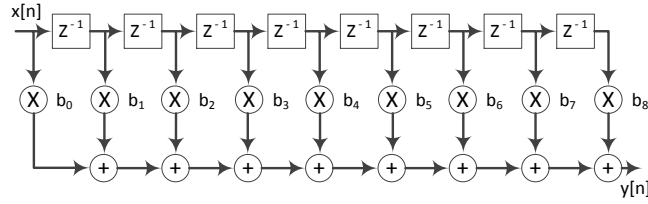


Figure 2.11: Data flow graph of a standard FIR filter with order K=8.

symmetric coefficients and depending on whether the filter has an odd or even order equation (2.31) can be rewritten as following:

$$y[n] = \sum_{k=0}^{(K-1)/2} h[k](x[n-k] + x[n-K+k]) \quad (\text{odd order}) \quad (2.32)$$

$$y[n] = \sum_{k=0}^{K/2-1} (h[k](x[n-k] + x[n-K+k])) + h\left[\frac{K}{2}\right]x\left[n - \frac{K}{2}\right] \quad (\text{even order}) \quad (2.33)$$

In equation (2.32) and (2.33), the impulse response $h[n]$ is symmetric and again K corresponds to the order off the filter. As it can be seen the number of multiplications is brought down to approximate the half by rewriting the equation. A DFG of equation (2.33) can be seen in figure 2.12. A DFG for the odd order folded FIR filter looks almost the same, just with a minor change in the delay-line. This folded FIR filter structure is a well-known strategy for implementing linear phase filters, and therefore seen as a good reference to measure the FBF model against.

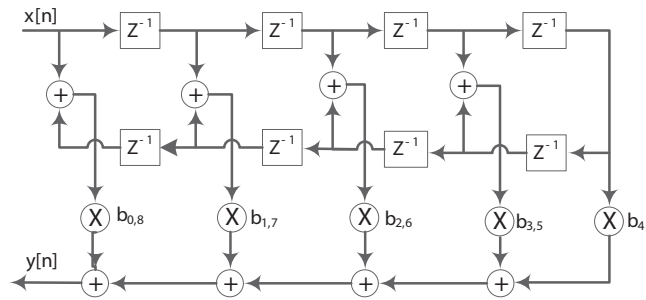


Figure 2.12: Data flow graph of an even order Folded FIR filter structure.

Chapter 3

Model Analysis

In this chapter it is evaluated when the FBF model start to take advantage in terms of a lower number of mathematical operations per sample. Further, a simulator of the FBF model is introduced, and to verify this simulator in a floating-point scenario the IIR filters in the FBF model is changed to finite length filters. Later the simulator is transformed to a fixed-point simulator using 2's compliments number representation where some different approaches have been used to verify the amplitude response and the phase response of the FBF model.

3.1 Mathematical Operations per Sample

In a real-time system a filter typically needs to process one sample through the filter to deliver one sample on the output, and the processing have to be finished before a new sample arrives. This means that the problem size will be fixed once the filter is designed, because this will specify the filter type and order. In this section the computational complexity will be evaluated based on mathematical operations per input sample. This is carried out since it is assumed that power consumption scales linear with the number of mathematical operations. In this section control overhead not is considered and it is assumed that a multiplication requires the same power as an addition. This is a very rough assumption, but is what we assume for this chapter.

3.1.1 Forward-Backward filtering model

The FBF model, which was designed in section 2.1, is evaluated here. Studying figure 2.9 for the FBF model and equation (2.30) it is possible to see that 3 identical filters and one adder is needed. The filters are specified as being Direct form 1 IIR filters with an order of K , and the problem size per sample of the FBF model can be stated as the following function.

$$FBF_{ops}(K) = \#filters(\#mul + \#add) = 3(2K + 1 + 2K) + 1 = 12K + 4 \quad (3.1)$$

The equation clearly shows that the function is linear, and it can be observed that problem size increase by 12 mathematical operation per unit increase in filter order.

3.1.2 Reference filter

The reference filter, which was designed in section 2.2, is evaluated here. First, some considerations on the odd order filter case, where an even number of coefficients is given, which was shown in equation (2.32). For the folded FIR filter of even or odd order the number of mathematical operations is then defined as follows:

$$FIR_{ops}(K) = \#mul + \#add = \begin{cases} \frac{K+1}{2} + K = \frac{3}{2}K + \frac{1}{2} & \text{if } K \text{ is odd} \\ \frac{K}{2} + K + 1 = \frac{3}{2}K + 1 & \text{if } K \text{ is even} \end{cases} \quad (3.2)$$

Looking at the odd or the even part as individual functions in equation (3.2) we are able to conclude that they are linear functions as the FBF model. This reference FIR filter structure has a positive linear slope equal to $\frac{3}{2}$ as the increase of mathematical operations per order increase.

3.1.3 Comparison based on filter order

After reading the two previous subsections one could easily think that the reference filter always is going to have the fewest mathematical operations, since the FBF model is having a slope of 12 and the slope of the reference filter is $\frac{3}{2}$. But one fact that has not been considered yet, is that the FBF model is using recursive filters, which meets a certain amplitude response by a much lower order. First some considerations for the general case where there is no thoughts given on how the filter order maps to a amplitude response specification. This makes it possible to see when either the reference filter or the FBF model should be used, when the considerations only is based on the number of mathematical operations. In figure 3.1 it is possible to

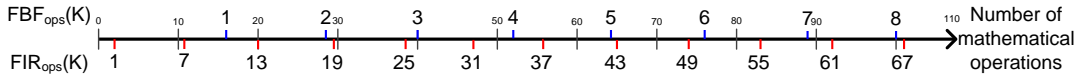


Figure 3.1: The black line has an interval of $[0;110]$ number of mathematical operations. The blue marks the mathematical operations needed to process the K'th order IIR filters used in the FBF model. Like-wise the red marks shows the number of mathematical operations for a K'th order folded FIR filter.

compare each of the filter designs against each other. Remember that it is assumed that the energy consumption is linear with the number of mathematical operations. Furthermore the control overhead is not considered either. An example could be that a 61th order FIR filter is needed but a 4th order IIR filter is needed to the FBF model to achieve the same amplitude response requirements. Using this information in figure 3.1 it can be seen that the FBF model has a lower number of mathematical operations per sample, and might therefore have a lower energy consumption than

the reference filter. Remember that it is the squared amplitude response of the IIR filter according to section 1.3 that should be evaluated to the design specification. Each amplitude response specification will lead to different filter orders for the two methods. The line in figure 3.1 can therefore be used as a guideline to figure out whether the FBF model is usable in the specific case.

The general case is set and evaluated according to the use of mathematical operations. Since there is no general guideline between a filter order and a design specification, a low-pass filter case is studied in appendix A where the transition band moves from wide to narrow. The specifications from table A.1 in appendix A is shown in figure 3.2 for reader convenience. The stopband frequency is stepping in 100 steps from a stopband frequency of 0.8ω to a stopband frequency of 0.11ω . This will force a more and more narrow transition band and thereby higher orders filter. It is also taken into account that the IIR filter only needs to have half the gain as

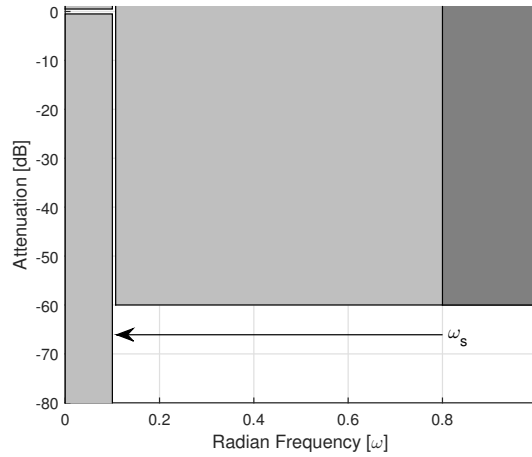


Figure 3.2: Random chosen design specification from table A.1, where ω_s moves in 100 steps from 0.8 to 0.11ω

explained in section 1.3. In appendix A an explanation of all the details on how the test is carried out is given. A simplified explanation of the test is as follows. Filter design parameters is set for the first case with 0.8ω as stopband frequency and filter order for both methods is found. The filter orders are found by using Matlab filter design package for Parks-McClellan optimal FIR filter as the reference filter. For the FBF model the lowest order elliptic IIR filters are found and used. These design methods have been chosen since they aim to give the lowest filter order. The Parks-McClellan optimal FIR filter design is found to be used by other Matlab functions that aims to find the lowest order FIR filter [9]. The elliptic IIR filter has ripple both in the passband and the stopband, and a narrow transition [13, p. 35]. These filter orders are then processed in equation (3.2) or (3.1), depending on which case there is calculated. The stopband frequency is now updated to be $1/100(0.8 - 0.1)\omega$

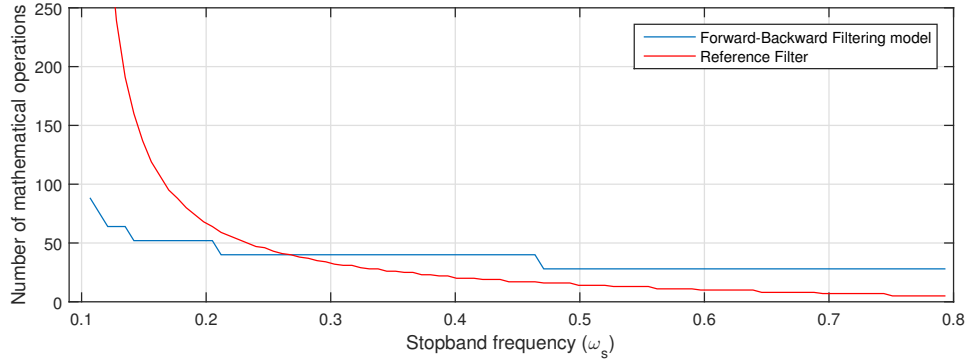


Figure 3.3: The plot shows the number of required mathematical operations which is needed to process the two implementations per sample.

closer to 0.1ω , and the filter order is found again. In figure 3.3 it is clear to see that a wide transition band results in a low order filter and for this case the reference filter will have the fewest mathematical operation. For the filter design with the narrow transition band it is clear to see that the FBF model is having the lowest number of mathematical operations. By looking at the plot both methods seem to have an exponential decreasing behavior but the reference filter seems to have a higher exponential value. The last point of the folded FIR is far outside the plot, because it has 734 mathematical operations. This is more than the 76 mathematical operations that the FBF model is requiring for the same amplitude response. The example is based on a rough estimate, but it is clearly seen when the FBF model are taking benefit of using IIR filters.

3.2 Model Simulation

A simulation of the FBF model from section 2.1 has been conducted and explained in appendix B. The model can be processed in two different cases due to the restriction of the block length L .

- A floating-point precision coefficients and signal values example with a non-linear phase FIR filter with a maximum order of $L-1$. A $L-1$ order FIR filter will have a impulse response of length L , such that the restriction of block length L is met. This is shown in section 3.2.1, and the purpose of this is to show that the model is performing as intended, as long as the restriction for the block length is met.
- A fixed-point precision coefficients and signal values example with a IIR filter. After analyzing the numerical aspects when implementing the FBF model in fixed point, section 3.4 discusses evaluation of the amplitude response.

The relation between these two cases gives rise to a discussion that will be carried out during the rest of this chapter. The floating-point evaluation aims to validate

the simulation against the theory, and the fixed-point aims to show the size of the error when the FBF model is executed under fixed-point constraints.

3.2.1 Floating-Point Simulation

To verify that the FBF model is functioning as intended, a test case has been designed. This test case have to comply with the restriction for the impulse response length for the filters in the FBF model. To secure that the used filters in the FBF model have non-linear phase a random low-pass IIR filter has been designed. However, a problem with an IIR filter in floating-point precision is the impulse response length is infinitely long. Instead the L length impulse response of an IIR filter is then used as coefficients to a FIR filter, to ensure that the filter has non-linear phase and a finite impulse response. The coefficients is determined as

$$b_{test}[n] = \sum_{i=0}^K b_i \delta[n-i] - \sum_{i=1}^K a_i b_{test}[n-i], \quad n = [0..L] \quad (3.3)$$

These coefficients $b_{test}[n]$ are then used in a FIR filter in the floating-point simulation of FBF model that have been described in appendix B. The FBF model is then fed

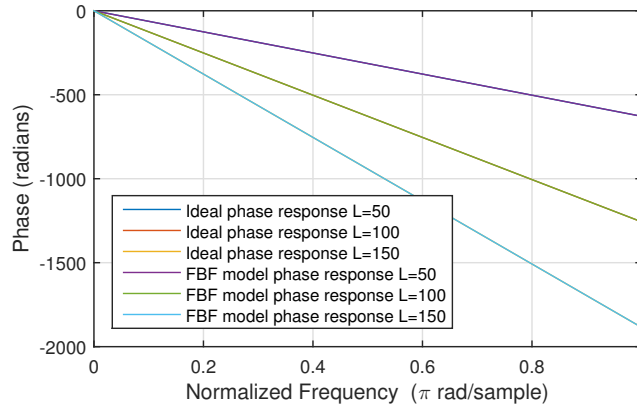


Figure 3.4: Phase response for the ideal phase response and the phase response of the FBF model. The simulation is done for 3 different lengths of L ; 50, 100 and 150. The FBF model is almost exactly on top of the ideal phase response, which is why they are not visible.

with an impulse to create the impulse response for the FBF model. The frequency response of the FBF model will then be the Discrete Fourier Transform (DFT) of the impulse response. In figure 3.4 the ideal phase response is almost exactly underneath the phase response of the FBF model. To make it more clear the error signal of the simulation has been found and shown in figure 3.5. As it can be seen the y-axis is small and it is assumed that the error occurs due to missing precision in the calculations of the frequency response. However, the non-linear filters in the

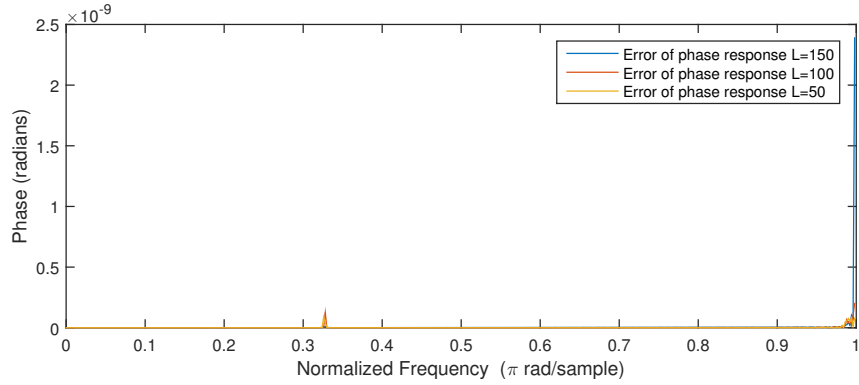


Figure 3.5: Absolute error of the simulated phase response for the FBF model with the use of non-linear phase FIR filters. The simulation is done for 3 different lengths of L ; 50, 100 and 150.

FBF model produces an almost linear phase response as result. The code for this simulation can found on the appendix-CD [./Matlab/FiniteImpulseEksample.m](#)

3.2.2 Numerical Aspects

As mentioned above, the target platform has fixed-point precision. Fixed-point precision leads to quantization of signal data and coefficients which is meet by rounding or truncation, such a binary description in 2's compliment is possible. As mentioned above, it has been chosen to use direct-form 1 IIR filters in the FBF model. This removes the possibility of transforming the IIR filters to cascaded second order sections, which in many cases is applied on fixed-point IIR filters with an order higher than 2. If second order filters were used, it would change the equation for the operations per sample in section 3.1.3. This section is based on materiel in [10]. The binary precision is described by the Q number format.

Coefficients Quantization

Quantization of coefficients is typically done by using the rounding method, and this introduces an error to the coefficients. This error will change size depending on the chosen binary precision, and this error will change the frequency response. The definition for quantizing the a and b coefficients to an IIR filter can be seen below.

$$a, b \cong Q_r[a, b] = \hat{a}, \hat{b} \quad (3.4)$$

Evaluating these quantized coefficients on the unit circle will define the new frequency response. This is a iterative process, since this new frequency response still have to comply with the initial specification. A discussion on how the actual amplitude response is found is given later in section 3.4.

Scaling

In scaling 3 common methods exist, maximum value scaling, sinus scaling, or variance scaling. Since no application is connected to the project, such that a conclusion could be determined through the applications restrictions, variance scaling has been chosen to be used. The variance scaling factor is given as follows.

$$s^2 \leq \frac{1}{\sum_{n=-\infty}^{\infty} |h_k[n]|^2} \quad (3.5)$$

$$s \leq \sqrt{\frac{1}{\sum_{n=-\infty}^{\infty} |h_k[n]|^2}} \quad (3.6)$$

The impulse response to the k^{th} node in the filter is defined as $h_k[n]$. The k^{th} node is defined by the critical point where overflow is not allowed. For the direct form 1 IIR filter the only critical point is on the output value. The scaling is applied

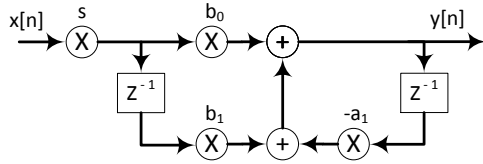


Figure 3.6: Scaling coefficient multiplied on the input value of a first order Direct form 1 IIR filter.

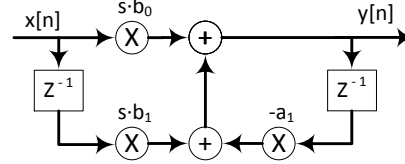


Figure 3.7: Scaling coefficient multiplied on the feed-forward coefficients of a first order Direct form 1 IIR filter.

to each of the individual filters in the FBF model. Instead of multiplying the scale factor on the input as in figure 3.6 the scale factor can be multiplied with the feed-forward coefficients as in figure 3.7. This secures that no extra multipliers for the implementation is needed.

Signal Quantization and Signal to Noise Ratio

For fixed-point platforms signal quantization is usually performed by truncating the variables. This method does not require any processing as e.g. the rounding method would do. A truncation simply keeps the most significant bits. A fixed-point implementation needs quantization since a multiplication of two Q.15 variables will result in a Q.30 result with 2 sign bits. This result can then be truncated to the Q.15 format by left shifting the result once and keeping the 15 most significant bits. In digital filters the multiplication is often followed by an addition with another result of a multiplication. This means that the Q.30 binary precision can be kept in the addition and wait with the truncation to after the final addition. This lowers the quantization error. The quantization error is seen as a wide-sense stationary white-noise process and has a uniform distribution of amplitudes over the quantization

interval. Each quantizer is uncorrelated with the input and all other quantization noise sources. The error for a truncation is defined in following interval.

$$-2^{-(N-1)} < e[n] \leq 0 \quad (3.7)$$

where N is the word length. Since the error can be seen as described, the mean and variance is defined as the following.

$$m_e = \frac{-2^{-(N-1)}}{2} \quad (3.8)$$

$$\sigma_e^2 = \frac{(0 - (-2^{-(N-1)}))^2}{12} = \frac{2^{-2(N-1)}}{12} \quad (3.9)$$

When the variance for the system is found it can be used to determine the Signal to Noise Ratio (SNR) for the system.

$$SNR = \frac{\sigma_y^2}{\sigma_e^2} \quad (3.10)$$

where σ_y^2 is the variance of the output. Since we want to scale the output to use the dynamic range from $[-1;1[$ we can calculate the variance for this as well

$$\sigma_y^2 = \frac{(max_y - min_y)^2}{12} = \frac{(1 - (-1))^2}{12} = \frac{1}{3} \quad (3.11)$$

For this project it is interesting to know the SNR for the FBF model and the reference filter. At this point the representation of the SNR will be found based on filters with N-bit multiplier and 2N-bit adders. As it can be seen in figure 3.8 this lead to one quantizer that is affecting the signal in the reference filter. This quantization is on the output and the SNR can therefore be calculated based on the variance of

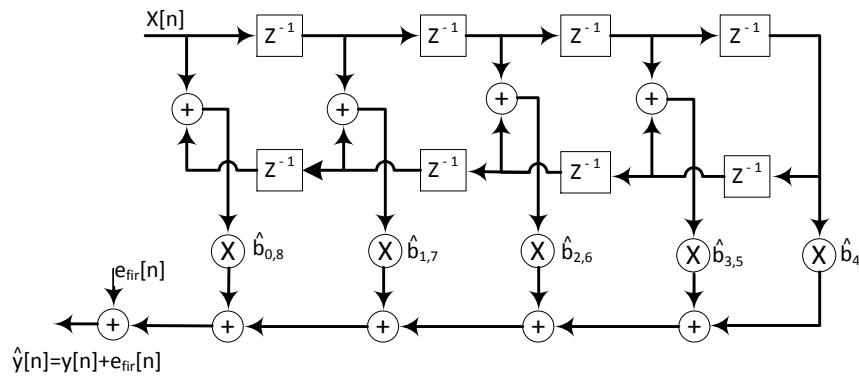


Figure 3.8: Data flow graph of the reference filter where the noise source is added which occur when the output is truncated.

one quantization error directly such $\sigma_e^2 = \sigma_{e_{fir}}^2$. The reference filter will have the following SNR.

$$SNR_{ref} = \frac{\sigma_y^2}{\sigma_{e_{fir}}^2} = \frac{4}{2^{-2(N-1)}} \quad (3.12)$$

The FBF model is more complicated since each filter generates noise, and this noise is filtered in the last filter. As seen in the previous chapter the direct form 1 IIR filter structure was chosen for the filters in the FBF model. An example of the FBF

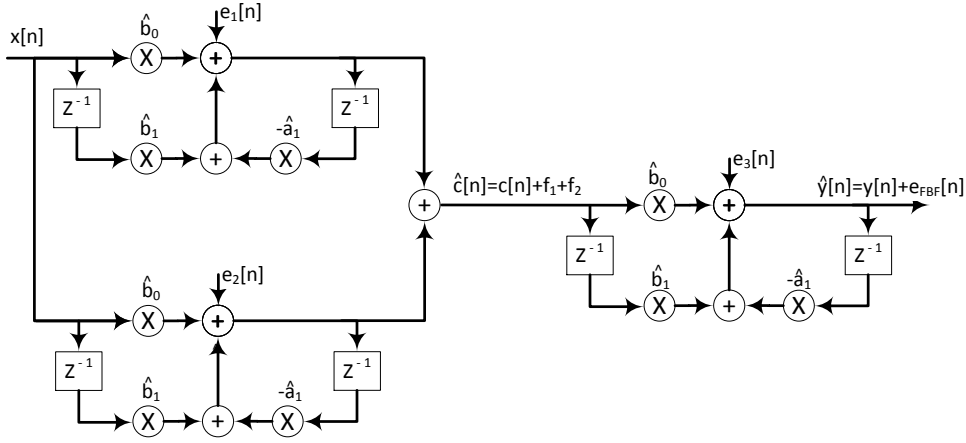


Figure 3.9: The error that occurs by the truncation noise is added in the model, where all switches and delays are removed.

model with identical first order direct form 1 IIR filters is seen in figure 3.9. In this example the buffers and the switches have been removed since they do not affect the statistical noise contributions. In a direct form 1 IIR filter the quantization error will occur where the feedback is added together with feed forward. The variance of noise from each of the individual identical IIR filters is then given by

$$\sigma_{f1}^2 = \sigma_{e1}^2 \sum_{n=0}^{\infty} |h_{ef}[n]|^2, \text{ where} \quad (3.13)$$

$$H_{ef}(z) = \frac{1}{1 - a^2} \quad (3.14)$$

As it can be seen in equation 3.13 a new impulse response $h_{ef}[n]$ occur. The impulse response $h_{ef}[n]$ is the impulse response from the point that the noise $e_1[n]$ occur in the individual filter to the output of the individual filter. The error on the output $f[n]$ is caused by the truncation error $e[n]$ in the individual filters. These errors are combined in the summation and the noise variances are therefore additive. Since it is 3 identical filters the variances are identical as well.

$$\sigma_{e1}^2 = \sigma_{e2}^2 = \sigma_{e3}^2 \quad (3.15)$$

$$\sigma_{f1}^2 = \sigma_{f2}^2 = \sigma_{f3}^2 \quad (3.16)$$

This also means that the noise variance $\sigma_{f_1}^2$ that each of the identical filters contribute with is identical. In figure 3.10 it can be seen that an error is added directly to the

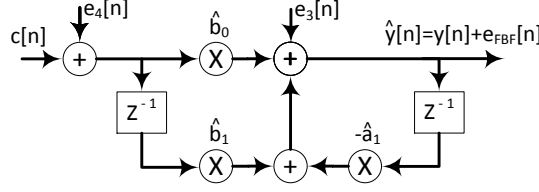


Figure 3.10: The error which is generated by e_1 and e_2 is here described by e_4 . The first two filter is removed since our concern is the error on the output.

input at the filter. This $e_4[n]$ is the error that the two filters on the left side in figure 3.9 generates. Since the noise sources are considered as white-noise and independent the variances are additive and defined as follows

$$\sigma_{e4}^2 = \sigma_{f1}^2 + \sigma_{f2}^2 = 2\sigma_{f1}^2 = 2\sigma_{f2}^2 \quad (3.17)$$

The noise-variance that originates from the input can then be calculated as below.

$$\sigma_{f4}^2 = \sigma_{e4}^2 \sum_{n=0}^{\infty} |h[n]|^2 \quad (3.18)$$

where $h[n]$ is the impulse response of the filter. This means that the noise of the output from the contribution can be calculated. These two variances are to be added together to calculate the total variance of the noise.

$$\sigma_{FBF}^2 = \sigma_{f4}^2 + \sigma_{f3}^2 \quad (3.19)$$

$$\sigma_{FBF}^2 = \sigma_{e4}^2 \sum_{n=0}^{\infty} |h[n]|^2 + \sigma_{e3}^2 \sum_{n=0}^{\infty} |h_{ef}[n]|^2 \quad (3.20)$$

$$\sigma_{FBF}^2 = 2\sigma_{e2}^2 \sum_{n=0}^{\infty} |h_{ef}[n]|^2 \sum_{n=0}^{\infty} |h[n]|^2 + \sigma_{e2}^2 \sum_{n=0}^{\infty} |h_{ef}[n]|^2 \quad (3.21)$$

The total variance σ_f^2 can than be compared with the variance of the output to calculated the SNR.

$$SNR_{FBF} = \frac{\sigma_y^2}{\sigma_{FBF}^2} \quad (3.22)$$

With this formula the SNR can then be computed for the FBF model when the precision is known.

3.3 Zero Input Limit Cycle Oscillation

As previous mentioned in section 2.1 the assumption for using IIR filters in the FBF model was the fact, that the impulse response of the an IIR filter will decay to a

value less than the precision used for the fixed-point platform, and hereby the output could be truncated to zero. Unfortunately this is not always the case on a fixed-point platform since the output can oscillate infinitely, due to quantization of signal values. For signal values the common quantization method is truncation, since it requires no additional logic to perform. Quantization can make a recursive filter suffer from what is known as zero input limit cycle oscillations. Zero input limit cycle oscillations is difficult to show in any general sense, but is better described by an example. A recursive filter with following difference equation is used as an example.

$$y[n] = \frac{5}{8}y[n-1] + x[n] \quad (3.23)$$

The recursive filter is chosen to be implemented as in figure 3.11. This changes the

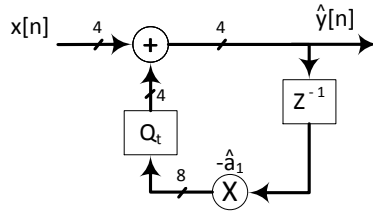


Figure 3.11: Data flow graph of equation (3.24).


recursive filter's difference equation to the one in equation (3.24) which suffers from zero input limit cycle when using truncation.

$$\hat{y}[n] = Q_t \left[\frac{5}{8} \hat{y}[n-1] \right] + x[n] \quad (3.24)$$

Coefficient, input and output variables are chosen to be 4 bit and Q_t is a 4 bit quantizer conducted by truncation. Applying an input $x_1[n] = -0.5\delta[n]$ or $x_2[n] = 0.5\delta[n]$ on

sample number [n]	0	1	2	3	4	5	6
Input $x_1[n]$	-0.5	0	0	0	0	0	0
Before truncation $y_1[n]$	-0.5	-0.31	-0.23	-0.16	-0.16	-0.16	-0.16
After truncation $\hat{y}_1[n]$	-0.5	-0.38	-0.25	-0.25	-0.25	-0.25	-0.25
Input $x_2[n]$	0.5	0	0	0	0	0	0
Before truncation $y_2[n]$	0.5	0.31	0.16	0.8	0	0	0
After truncation $\hat{y}_2[n]$	0.5	0.25	0.13	0	0	0	0

Table 3.1: This shows the input/output values in two cases where of zero input limit cycle occurs in one of the cases. The individual columns represent consecutive time to sample number n.

the recursive filter in equation (3.23) can be seen in table 3.1 and figure 3.12 and the script for this can be found on the appendix-CD  ./Matlab/LimitCycleExample.m.

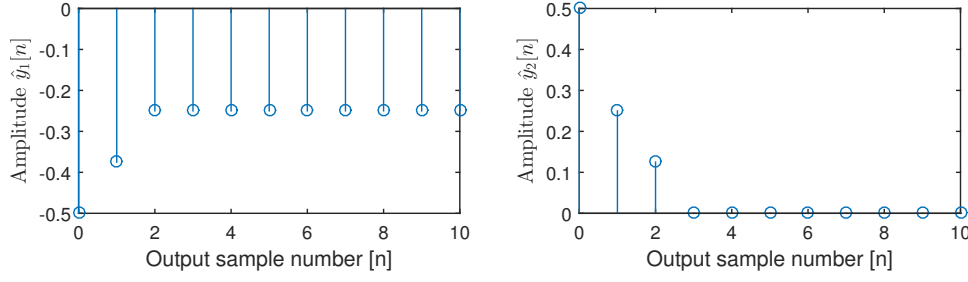


Figure 3.12: Plot of output for $\hat{y}[n]$ from Table 3.1 in both types of input.

It can clearly be seen that the output $\hat{y}_1[n]$ keeps being in a steady-state at -0.25 since it keeps getting truncated to this value as long the input is exactly zero. This happens since truncation is a quantization method that always round against negative infinity. Looking at the positive impulse $x_2[n] = 0.5\delta[n]$ it can be seen that output is truncated to zero after 3 samples. This meanw that the same recursive structure can make some signal end in a zero input limit cycle while other may avoid it. Further, each recursive structure will generate each their type of zero input limit cycle oscillation when using truncation as the quantization. If the quantization method was changed to e.g. rounding the zero input limit cycle response might change again, but for low power implementations truncation is often chosen as discussed earlier.

When zero input limit cycle oscillations occur the filter might no longer fulfill the stability requirement:

$$\sum_{n=-\infty}^{\infty} |h[n]| < P \quad (3.25)$$

The stability requirement in equation (3.25) tells that the sum of the absolute impulse response must be below a certain upper boundary P. Theory can be found on how to design IIR filters which do not suffer from zero input limit cycle but this is not seen as the focus for the project. Instead we assume that there will be a non-zero signal applied to the input at all times. In that case the system will only generate a zero input limit cycle oscillation in the reverse filtering part, if the length of the block is too long or the signal variation is low. The effect of zero input limit cycle oscillation is unwanted. Therefore it is important that the Block length is no longer than the actual impulse response for the final system.

3.4 Amplitude Response Discussion

In this section it is discussed how the amplitude response of the FBF model is found. Lets recall equation (2.27) showing the derived amplitude response from section 2.1.2.

$$|\hat{H}_{FBF}(e^{j\omega})| = |\hat{H}(e^{j\omega})|^2 \quad (3.26)$$

Equation (3.26) tells that the used filters results in the squared amplitude response for the FBF model, but this is based on following. It was initially assumed that

when the output of the filter was below the used resolution the output would be zero, and the infinite impulse response filter in that case would be finite due to missing precision. But as it was illustrated in the zero input limit cycle oscillation example in section 3.3, this is not always the case. Before discussing the fixed-point effects of this system a small analysis of the FBF model with quantized coefficients using floating-point variables is carried out.

The effect of changing L is not possible to see in the standard method to calculate the frequency response. If the reverse filtering IIR filter is named H_{rev} , the forward filtering IIR filter H_{for} and the block length L following can be derived.

$$\hat{H}_{FBF}(\omega) = \hat{H}_{rev}^*(\omega) \hat{H}_{for}(\omega) e^{-j\omega 4L} \quad (3.27)$$

$$|\hat{H}_{FBF}(\omega)| = |\hat{H}_{rev}^*(\omega) \hat{H}_{for}(\omega)|, \text{ where} \quad (3.28)$$

$$\hat{H}_{rev}[n] = \sum_{n=0}^L \hat{h}[n] e^{-j\omega n}$$

This will clearly conflict with the original fact that the amplitude response is the squared amplitude response. As equation (3.28) states the time reversed impulse response is truncated after L samples. The problem when the length of L is truncated, is the fact that the model no longer is LTI. It is no longer LTI since the response will change depending on when for instance an impulse enters. It happens if the impulse enters in the start or the end of a block such the reverse filtering will give different lengths of impulse responses. In figure 3.13 the impulse and the impulse response is shown when it arrives at two different samples in the block. Figure 3.13a shows the impulse response arrive as the first sample in a block. This creates the shortest possible impulse response with L samples to the output of the time reversed filtering part. Figure 3.13d shows the impulse arrives as the last sample in the block. This creates the longest possible impulse response of $2L-1$ samples. Comparing figure 3.13c and 3.13f it can be seen that the impulse response will have a different length depending on where in the block the impulse appeared. The biggest possible error to the FBF model must then be found to the impulse response for a impulse entering as the first sample in a block. Using the simulator from section 3.2.1 an impulse can be applied to the first sample in a block and an impulse response \hat{h}_{FBF} of the FBF model can be generated.

$$|\hat{H}_{FBF}(\omega, L)| = |DFT(\hat{h}_{FBF}[n])| \quad (3.29)$$

Hence the amplitude response can be found for the system by calculating the DFT of the impulse response.

Another method to choose the block length is proposed in [11, p. 2431]. Introducing impulse response that is truncated after L samples, the error that we introduce

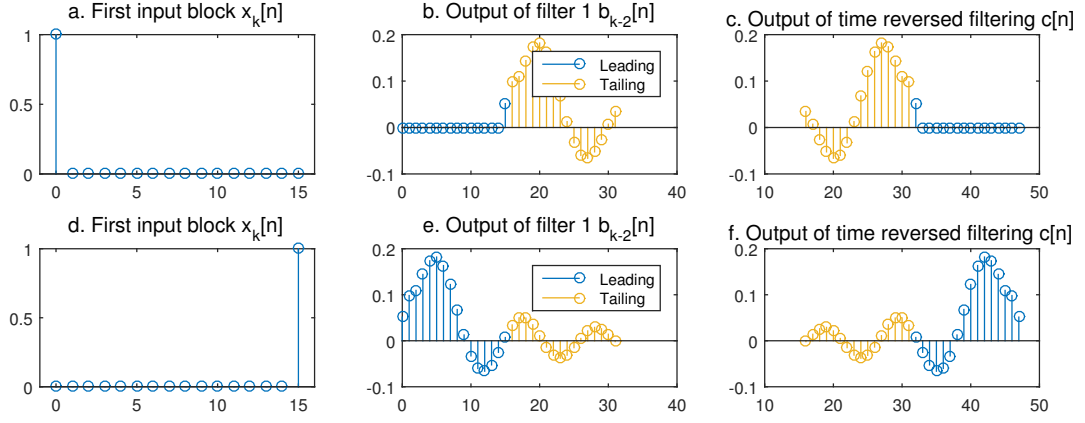


Figure 3.13: Illustration of an impulse applied to the reverse filtering part of the FBF model where an IIR filter is used. the block length L is set to 16 to make it easy to see that the impulse response is different for the two cases. In the first case in figure a, b, and c the the impulse is set for sample 0. In the other case the impulse is set for sample 15.

can then be defined as:

$$\hat{H}_{Error}(\omega) = \sum_{n=L}^{\infty} \hat{h}[n] e^{-j\omega n} \quad (3.30)$$

$$|\hat{H}_{Error}(\omega)| = \sum_{n=L}^{\infty} |\hat{h}[n]| |e^{-j\omega n}| = \sum_{n=L}^{\infty} |\hat{h}[n]| \leq \epsilon(L) \quad (3.31)$$

The maximum error to the amplitude response of the time reversed filter in the FBF model can be defined as in equation (3.31). According to [11] this maximum error is defined as the pass-band ripple δ_p or the size of the allowed stopband δ_s , depending on which is smaller. But theoretically this truncation of the signal in the reverse filtering is adding ripple to the amplitude response of the FBF model. In

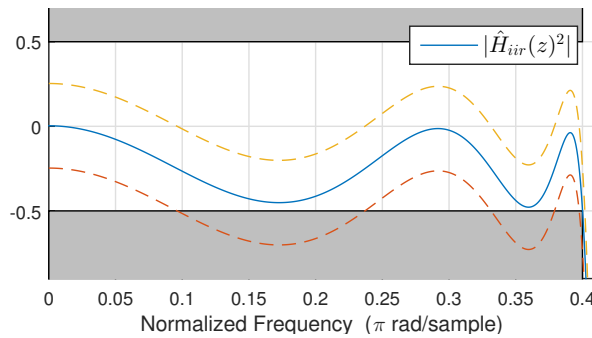


Figure 3.14: The squared amplitude response of an IIR filter. The dashed lines mark the possible errors by using the method given in equation (3.31).

figure 3.14 an example of a squared amplitude response $|\hat{H}_{iir}(z)|^2$ is shown and the

pass-band ripple δ_p yields the smallest error. Fitting L according to equation (3.31) such it complies with δ_p the amplitude response of the FBF model will be within the dashed borders in the figure. A method to secure that a design complies with the design parameter will then be to define the maximum band-pass ripple as $1/3\delta_p$ to each of the IIR filters $|\hat{H}_{iir}(z)|$. This will secure the amplitude response to comply

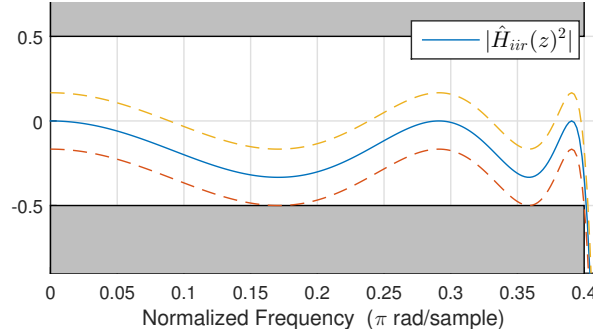


Figure 3.15: To secure the amplitude response to comply with the overall amplitude specification.

with the amplitude requirements. This method needs more narrow filter design and is therefore not seen as an optimal choice. For this project we will examine the amplitude response by applying an impulse response for the first sample in a block to the simulator as explained earlier in this section. If the block size is lowered it will lower the need for temporary variables but also the group delay since this is bounded to the block size.

Since the model only generate a linear phase response due to the effects of fixed-point numbers representation the correct phase response cannot be determine derived in the floating point simulator.

3.5 Phase Response Discussion

Until now the group delay has only been verified based on the theory in chapter 2 and by determining a non-linear phase FIR filter and using it for the floating point model as seen in section 3.2.1. But how is the phase error measured when using IIR filters? And how does the phase response react if the block length is made smaller?

Previously in this discussion the amplitude response of the FBF model was calculated for a low-pass filter. This means that we need precision for all bands which we have in an all-pass filter. Therefore a random all-pass filter has been designed and implemented in fixed point with 16 bit precision. The block length L is found by investigating the length of the impulse response before it enters a zero input limit cycle oscillation. This filter is then used in the FBF model to create a new impulse response. The Phase response of this impulse response is then subtracted from the theoretical as it is done in section 3.2.1. The absolute error of this is found and

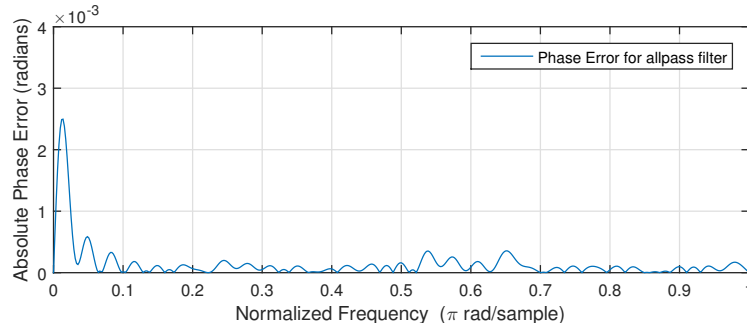


Figure 3.16: Absolute error of the phase for a all-pass IIR filter used in the FBF model with 16-bit fixed-point arithmetic. By visually inspecting the impulse response, The block length L was found to be 62 before entering zero input limit cycle oscillation.

shown in figure 3.16.

In the previous test case with the amplitude response we tried to only use the half block size. The same trick is now applied to this test case with the all-pass filter. In figure 3.17 the test we see the result of this scenario. The maximum error has not

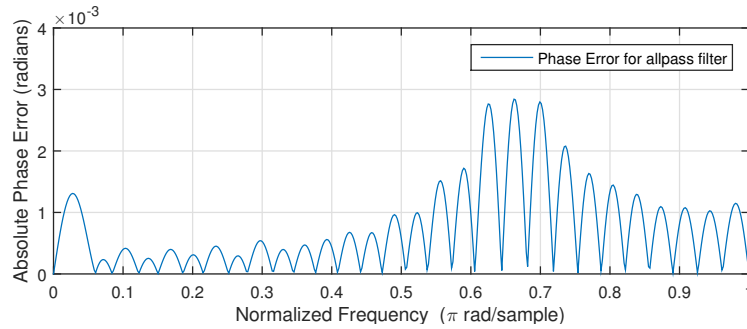



Figure 3.17: Absolute error of the phase for a all-pass IIR filter used in the FBF model with 16-bit fixed-point arithmetic. Block length of 31 samples was used.

changed more than from 0.0025 to 0.0029. This test can be found as a Matlab script on the appendix-CD  ./Matlab/PhaseErrAllpass.m.

3.6 Group Delay Considerations

In section 2.1.2 it was assumed that all filters were able to consume and produce one output per sample to the FBF model. But some further considerations can be made on the time reversed section and the related LIFO buffers. If it is assumed that infinite computing power is available, it is possible to lower the group delay. The group delay was defined by the $2L$ delay line in the model and the 2 LIFOs in the system that together made a $4L$ sample group delay to the FBF model. Studying

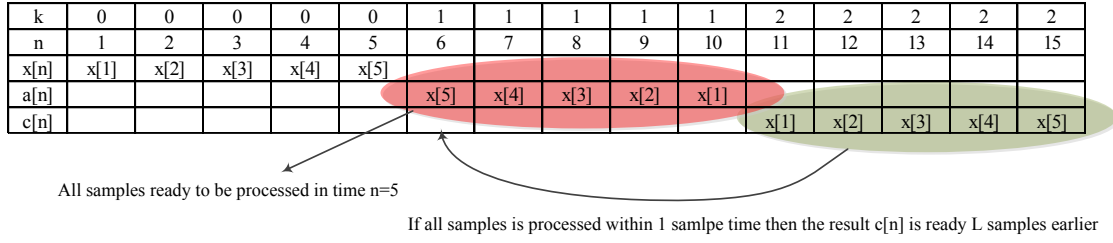


Figure 3.18: Timing schedule for $x[n]$, $a[n]$ and $c[n]$ with a block length L equal 5. The timing diagram is based on that $b_k[n] = a_k[n]$. The individual columns represent consecutive time to sample number n .

figure 3.18 and knowing the relation between $x[n]$ and $a[n]$ is the LIFO buffer, it can be seen that all samples from the first block are ready to be processed when sample number $n=5$ has arrived. If $a[6-10]$ is processed when they are available at sample

k	0	0	0	0	0	1	1	1	1	1
n	1	2	3	4	5	6	7	8	9	10
x[n]	x[1]	x[2]	x[3]	x[4]	x[5]					
a[n]					x[1-5]					
c[n]						x[1]	x[2]	x[3]	x[4]	x[5]

Figure 3.19: Timing schedule for $x_k[n]$, $a_k[n]$ and $c_k[n]$ with a block length L equal 5. The fast computation of $a_k[n]$ used to generate $c_k[n]$ faster. The individual columns represent consecutive time to sample number n .

time $n=5$, the result for $x[1]$ is ready in $c[6]$ instead of $c[11]$. This can be seen in figure 3.19 that the group delay that originates from the LIFO registers now has been reduced by L samples. This also means that the needed number of registers can go down if hardware sharing is utilized. In figure 3.20 the two time reversal blocks that was seen in chapter 2 have been reduced to one. When L samples have shifted in the buffer the data can be processed in the red areal without waiting for new samples since the next is ready in the buffer. The result is shifted in on the right side and a new sample is ready right away on the left side. The blue area is still processing 1 sample per input sample on $x[n]$. This means that the throughput for the model is identical but the group delay has changed.

If no constraints are given to the group delay and this is made larger again, some other observations can be made. If the block size L is made larger than what L originally was chosen to be, the tailing filter can be turned off such the mathematical operations for this filter is no longer needed. In figure 3.21 scenario A is the case that is documented in chapter 2, where H_{top} and H_{bot} are describing the two identical filters in the time reversed section. It can be seen that the H_{top} and H_{bot} alternately are computing the leading and tailing response. The block length for the used filters is found to 50 samples. This makes the filters active at all times. Scenario B illustrate

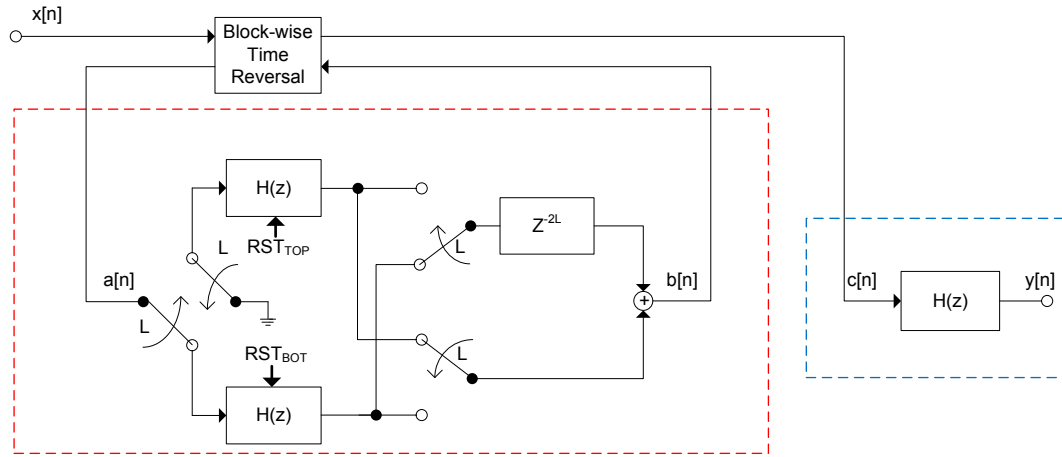


Figure 3.20: Revised model of the FBF model that takes advantage of the observation in figure 3.19.

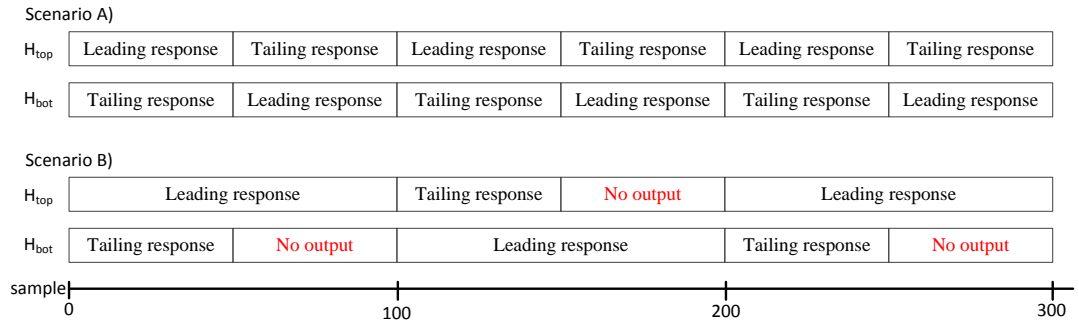


Figure 3.21: Output of the two filters in the time reversed section illustrated as blocks. The sample line represent consecutive time to sample number n

a block length is set to 100 samples but the used filters are still the same as in scenario A. This means that the tailing filter can be shutdown after 50 samples. This length will be defined as J samples. As it can be seen in figure 3.21 the tailing filter can be shutdown 50 % of the time. The cost of utilizing this is larger buffers and the larger group delay mentioned above.

Chapter 4

Fixed-Point Filter Design and Simulation

In the following chapter, requirements which both the reference filter and the FBF model have to fulfill are presented. The reference filter is a linear phase FIR filter and the FBF model is realized using a suitable IIR filter. Throughout the chapter these filters will be simulated as if they were running on a fixed point platform, to make sure that the requirements are met.

4.1 Design Templates

A template simply specifies which requirements a final design of a filter has to comply with. Since this project does not rely on any application, all specifications will be specified on a normalized frequency scale. This template defines the amplitude response requirements, for a low-pass filter template. Based on the knowledge which

	Low-pass Filter
Bandpass Ripple [dB]	0.5
Bandstop Attenuation [dB]	60
Transition band [ω]	0.05
Passband frequency [ω_p]	0.1

Table 4.1: Filter design parameters for each of the four templates.

was gained section 3.1.3 some design parameters were chosen. The stopband frequency is to be 0.15ω since it can be seen that the FBF model will have fewer mathematical operations per sample for a low-pass filter with this amplitude specification. The design template is then given as it can be seen on figure 4.1. These design parameters for the amplitude response is used to design filters for the FBF model and the reference filter.

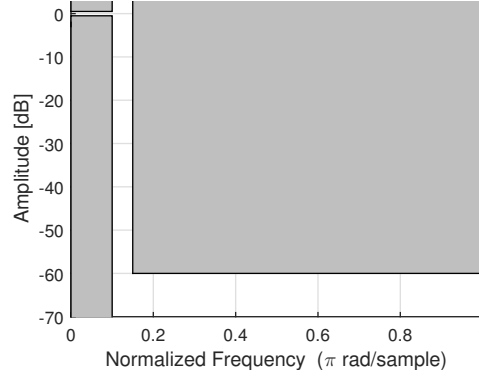


Figure 4.1: A plot of the filter design parameters in table 4.1.

4.2 Filter Design for the Forward-Backward Filtering Model

As seen in section 2.1.2 the model consist of 3 identical filters. As seen above these filters do only need half the attenuation for the overall model. It has been chosen to use elliptic IIR filter design for the 3 identical filters in the FBF model. This filter type allows a narrow transitions band since the filter allows ripple in both the transition- and pass-band [14, p. 602]. The filters are designed using the Matlab design function since the design process is not the scope of the project. After validating the quantized coefficients the filters quantization noise will be found. The requirements in table 4.1 led to a 4th order elliptic IIR filter $H_{iir}(\omega)$ that complies with the half ripple and stopband attenuation. The upper boundary scale factor is found according to equation (4.2). If this upper boundary scale factor is multiplied directly on the feed-forward coefficients, and the coefficients are rounded up when quantized for the 2's compliment notation to the fixed-point platform, it will no longer fulfill equation (4.1).

$$s \leq \sqrt{\frac{1}{\sum_{n=-\infty}^{\infty} |h_{iir}[n]|^2}} \quad (4.1)$$

$$s = \frac{\left\lfloor \sqrt{\frac{1}{\sum_{n=-\infty}^{\infty} |h_{iir}[n]|^2}} \cdot 2^{17} \right\rfloor}{2^{17}} = 3.024 \quad (4.2)$$

$$Q_r[s \cdot b] = \hat{b}_{upd} \quad (4.3)$$

$$s_{upd} = 1 \leq \sqrt{\frac{1}{\sum_{n=-\infty}^{\infty} |\hat{h}_{iir,upd}[n]|^2}} = 1.0001 \quad (4.4)$$

It has been chosen to calculated the scale factor as in equation (4.2) such that the value is close to the boundary. After multiplying this scale factor to the feed-forward coefficients and quantizing all coefficients by rounding, equation (4.1) can

be calculated again as seen in equation (4.4). The scale factor s is now set to 1 such we can conclude whether the scaled filter have a proper scaling according to variance scaling. It can be seen that the scaled filter $\hat{H}_{iir,upd}(\omega)$ still complies with the boundary. In the rest of the report the scaled and quantized filter will have transfer function $\hat{H}_{iir}(\omega)$.

4.2.1 Fixed-Point Simulation of Forward Backward Filtering Model

As mentioned 18-bit multipliers are available on the FPGA, these are intended to be used in full for the IIR filters. This sets the restriction that the coefficients for the filter cannot be described by no more than 18-bit. As seen in appendix C a fixed-point 4^{th} order IIR filter has been designed for the simulator from appendix B. The amplitude response of the elliptic IIR filter is shown in figure 4.2, with both the

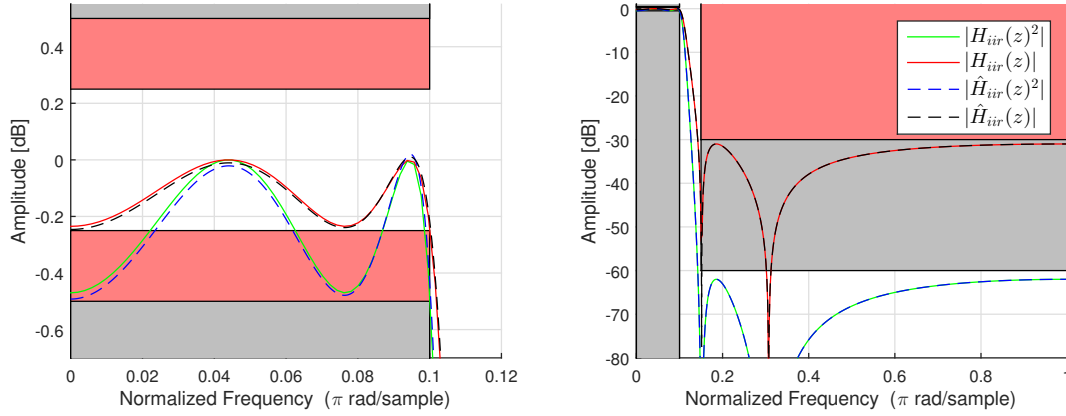


Figure 4.2: Amplitude response for low-pass 4^{th} order IIR filter where the grey shows the template and the red marks the required parameters for IIR filter used in the FBF model. The filter $H_{IIR}(z)$ is the result of the floating pint filter design coefficients, while $\hat{H}_{IIR}(z)$ is the scaled and quantized coefficients.

full precision coefficients and the scaled and quantized coefficients. The amplitude response of the scaled and quantized filter coefficients are normalized by the scaling factor, such it is possible to compare the two designs according to the requirements. The squared amplitude response shows that the overall amplitude response for the FBF model complies with design template in figure 4.1. The SNR is then found by

using equation (3.22) that was derived for the FBF model in section 3.2.2.

$$\sum_{n=0}^{\infty} |h_{iir}[n]|^2 = 0.999 \quad (4.5)$$

$$\sum_{n=0}^{\infty} |h_{iir,ef}[n]|^2 = 6370.2 \quad (4.6)$$

$$\sigma_{e2}^2 = \frac{2^{-2(N-1)}}{12} = \frac{2^{-2 \cdot 17}}{12} = 4.8506 \cdot 10^{-12} \quad (4.7)$$

$$\sigma_{FBF}^2 = 2\sigma_{e2}^2 \sum_{n=0}^{\infty} |h_{ef}[n]|^2 \sum_{n=0}^{\infty} |h[n]|^2 + \sigma_{e2}^2 \sum_{n=0}^{\infty} |h_{ef}[n]|^2 \quad (4.8)$$

$$SNR_{FBF} = 20 \log_{10} \left(\frac{\sigma_y^2}{\sigma_{FBF}^2} \right) = 301.9 \text{ dB} \quad (4.9)$$

As it can be seen the SNR-value is approximately -302dB. This error is an expression for the error in compare to the infinite precision calculation of data that has been filtered in the filter. Often signals originates from a sampled analog signal. When a analog signal is sampled to a digital signal in an 18-bit ADC the SNR the is defined as 6.02dB per bit for a uniform distribution. This result in a SNR of $18 \text{ bit} \cdot 6.02 \frac{\text{dB}}{\text{bit}} = 108.36 \text{ dB}$. This means that if the input signal exist from a ADC the noise that the filter generates will be lower than the noise that already exist in the signal.

The block length effect that was discussed for the amplitude response is now applied. First the block length L was solved by using the method proposed by [11] shown in equation (3.31). That resulted in a block length of 111 samples. Processing amplitude response of the FBF model as discussed in section 3.4 lead to the result that is seen in figure 4.3. The block length is afterwards iteratively made smaller to examine how short the block length can be and the amplitude response still complies

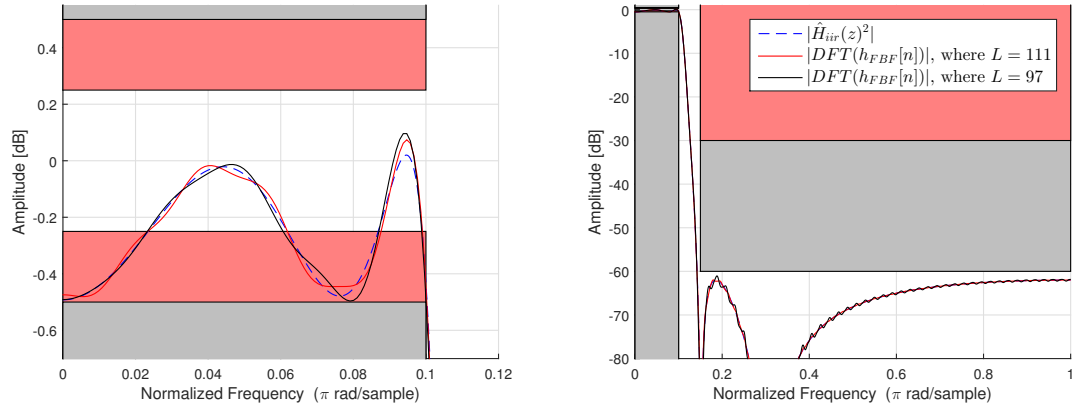


Figure 4.3: The filter $\hat{H}_{iir}(z)$ is the quantized, scaled and squared amplitude response from figure 4.2. The Black and red marks the DFT of the impulse response generated by the FBF model simulator in floating point.

with the specification. This block length was found to be 97 samples as it also can be seen in figure 4.3. It is also important that the block length is not made too long. This might result in zero input limit oscillation added to the signal. Appendix D shows the impulse response of the IIR filters that is designed in the section. A Matlab script generating the results in this section can be found on the appendix-CD `./Matlab/FinalFPLPFBF.m`

4.3 Design of Reference Filter

In this section the coefficients for the reference filter which is a folded FIR will be found. These coefficients are found by using the Parks-McClellan optimal FIR filter design function in Matlab. The scaling is performed by variance scaling as also was done to feed-forward coefficients in the direct form 1 IIR filters. The scaling parameter for the FIR filter $H_{ref}(\omega)$ is found.

$$s = \frac{\left\lfloor \sqrt{\frac{1}{\sum_{n=-\infty}^{\infty} |h_{ref}[n]|^2} \cdot 2^{14}} \right\rfloor}{2^{14}} = 2.997 \quad (4.10)$$

$$s_{upd} = 1 \leq \sqrt{\frac{1}{\sum_{n=-\infty}^{\infty} |\hat{h}_{ref,upd}[n]|^2}} = 1.000001 \quad (4.11)$$

The transfer function to the new scaled and quantized reference filter coefficients will be denoted $\hat{H}_{ref}(\omega)$.

4.3.1 Fixed Point Simulation of The Reference filter

In this section the linear phase FIR filters will be designed such that they meet the design template from section 4.1. The coefficients are quantized to a 18-bit wordlength. In appendix C a fixed-point 89th order FIR filter has been designed for the simulator. The amplitude response of the filter is shown in figure 4.4. It can

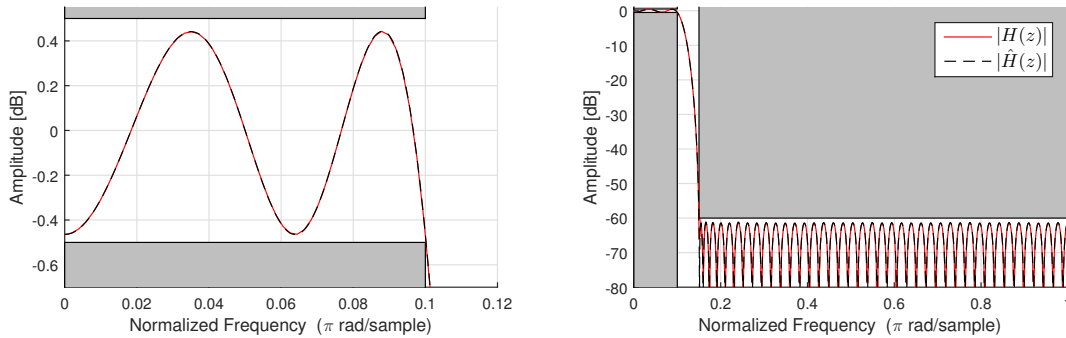


Figure 4.4: Amplitude response of the 89th order FIR filter.

be seen that the quantized filter is almost on top of the filter with the full precision coefficients. Again the frequency response of the scaled and quantized coefficients is normalized by the scaling factor to compare it to the non-scaled and quantized and the requirements. Another important concern is of course that the filter has linear phase. The coefficients are found to be symmetric thus the phase response should be linear. In figure 4.5 The phase response is shown. As it can be seen the phase is

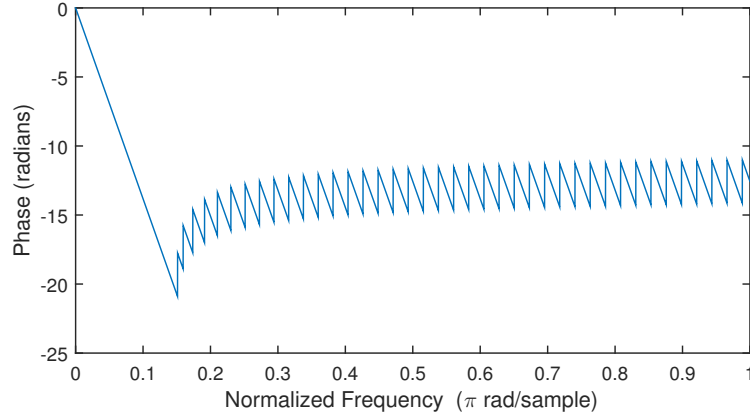



Figure 4.5: Phase response of the FIR filter.

linear until it moves into the stopband. At this point the phase starts to jump by π radians, which we ascribe to incorrectness of the phase evaluation of the filter in Matlab. It is verified that the filter coefficients are symmetric.

The SNR of this filter is found by using the same methods as for the FBF model. The variance of the output divided by the variance of the noise on the output. In this filter only one quantization exists directly on the output as seen in section 3.2.2. This quantization leads to the variance in equation (4.12).

$$\sigma_{ref}^2 = \sigma_{e2}^2 = \frac{2^{-2(N-1)}}{12} \quad (4.12)$$

$$SNR_{ref} = 20 \log_{10} \left(\frac{\sigma_y^2}{\sigma_{ref}^2} \right) = 499dB \quad (4.13)$$

This also means that a higher SNR than the FBF model is achieved as seen in equation (4.13). A Matlab script generating the results in this section can be found on the appendix-CD  ./Matlab/FinalDFIRLP.m

Chapter 5

System Design

In this chapter the FBF model and the reference filter is mapped from algorithm to architecture. The designed will be given as a RTL design for both models. This process is conducted by following some initial considerations for energy consumption. Theory in this chapter is based on [4].

5.1 Design Constraints and Abstractions Levels

In order to keep the synthesis process of the two algorithms in a structured manner, the methodology used is clarified in this section. In figure 5.1 the Y-chart is shown. The three axes represent the three aspect in every design, which again is divided into four abstraction levels. The behavioral aspect describes the system in terms of I/O

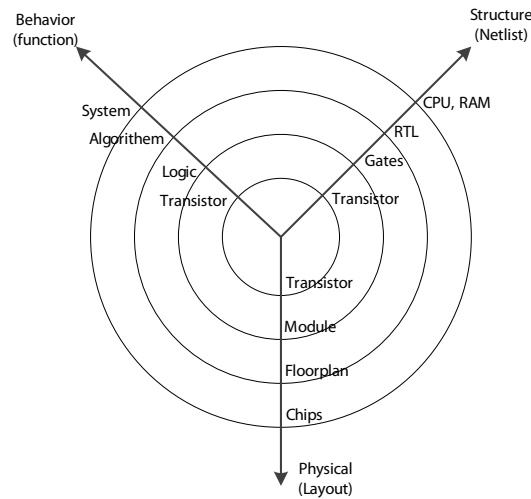


Figure 5.1: The figure shows what i known as the Y-chart, and the three axis describe the three aspect in every design.[4, p. 3]

relation, typically given by a function. The structure defines the system in terms of hardware and how they are connected. The physical aspect defines the size, shape, physical placement, and connections. The four abstraction layers generally represent the System, Processor, Logic, and Circuit layer. Each layer can then be defined at each aspect. Different methodologies are available to move through the Y-chart. In this case the Platform is known to be an FPGA, and synthesis of the system layer from System to CPU, RAM is therefore already accomplished. Knowing the hardware the next step is the synthesis of the processor layer from Algorithm to RTL description. For this project this is conducted by synthesizing the algorithm in the following order. First a Precedence Graph (PG) is constructed from the DFG. Time is introduced to the PG in order to do cycle-accurate scheduling. First task is to bind the variables in the PG to different registers, and second the operations are bounded to the Functional Unit (FU)s such as a multiplier or an adder. The registers and FUs are then connected in a structure such the PG can be processed as scheduled. Afterwards a controller is conducted in order to execute the scheduling defined in the PG. This can be a recursive process since one can discover more optimized schemes in the synthesis process, which means that one could need to move back and start over. All considerations for this synthesis are made with energy savings as the main goal.

5.2 Cost Function

Mapping from an algorithm to an architecture is a one-to-many solutions task. Each solution will have different use of area, power, precision and time. Each of these parameters are put together in what is known as the cost function. The cost function is a tool to find the optimal solution when performing this mapping. Typically some constraints are set for the timing and the architecture. The goal is then to minimize the cost with the constraints that are set. In this project the two models are implemented using as low energy as possible. At this point a few constraints to the some parameters in the cost function are set.

- It was chosen in the design phase of the filters to use the full 18-bit precision of the embedded multipliers on Altera FPGA board.
- The time is limited to the time interval between two incoming samples from the ADC given by $t_{samp} = \frac{1}{f_{samp}}$, where f_{samp} is the sample rate for the input signal.
- There have been given no boundaries to the area, but the project is limited to use no more than the 35 embedded multiplies in the FPGA. The FPGA also have 33216 LEs to create the architecture that will be design and presented during this chapter.
- For this project energy consumption is the parameter to minimize. Since energy consumption is power integrated over time, the power is a focus parameter as

well. The power usage is given by the following equation for the dynamic related power consumption in transitions.

$$P = C_L V_{dd}^2 f_{clk} \quad (5.1)$$

Where C_L is the capacitance, V_{dd} is the supply voltage, and f_{clk} is the system clock frequency.[1, p. 130] But the power is not constant, since it relies on which and how many FUs that are processing at any given time.

Mapping an algorithm to an architecture is an one-to-many solutions task. Depending on the number of FUs that are chosen to be assigned, the algorithm can be processed with a different amount of clock-cycles. When the amount of FUs goes up the processing time most likely go down due to use of the possible inherent parallelism. A trade-off between processing time and the area consumption needs to be found. Using the boundaries from the cost function, a solution needs to be found as shown in figure 5.2. For this project we want to suggest a design with the lowest

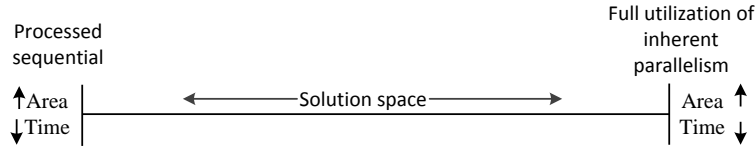


Figure 5.2: The trade-off between the use of area and time to process an algorithm on an architecture.

possible energy consumption. Considering above solution space, some conclusions to the implementation can already be made. If it is chosen to reuse hardware, some logic need to control which inputs are selected and which register is selected for the result. This addition of control logic will lead to an increase of energy consumption. By the given limitations a conclusion to minimizing the cost function, will be to implement as a one-to-one mapping or as close to that as possible. A one-to-one mapping means that the operations on the DFG are assigned to the registers and FUs in the same structure.

5.2.1 Supply voltage

For this project it is assumed that the used logic, is working at its critical rise time t_{rise} or fall time t_{fall} , when operating at 50 MHz. An FPGA can work with much higher frequencies but since the goal is an ASIC design it is assumed that the FUs that are used, have a critical rise or fall time at 50 MHz when operating af 1.2 V. Since voltage is a squared parameter to the power usage in equation (5.1), it is a very useful parameter to lower in order to get lower energy consumption. When the voltage is lowered the rise and fall time is changed approximately according to the

following equation. [1, p. 127]

$$t_{rise} \approx k \frac{1}{V_{dd}} \quad (5.2)$$

where k is a value which depend on the used hardware. It is assumed that constant k is causing the critical rise or fall time. For this project we assume that it has been possible to lower the voltage to 1V before the electronics simply started to deliver wrong results on the output. When lowering the voltage the system clock frequency lowers according to

$$f_{clk,new} = f_{clk,old} \frac{\frac{1}{V_{dd,new}}}{\frac{1}{V_{dd,old}}} = 50 \frac{1}{1.2} = 41.67\text{MHz} \quad (5.3)$$

With lower voltage the maximum clock frequency is now found to be 41.67 MHz. The voltage is now lowered such that a lower energy consumption is achieved. Both the FBF model and the reference filter will be implemented with this new system clock frequency assigned reassigned to f_{clk} . With this lower frequency the power usage per sample has been lowered according to following.

$$1 - \frac{C \cdot V_{dd,new}^2}{C \cdot V_{dd,old}^2} = 1 - \frac{V_{dd,new}^2}{V_{dd,old}^2} = 0.306 \quad (5.4)$$

This results in a 30% lower energy consumption per clock pulse to the same amount of hardware. Now that the system clock frequency has changed, the number of cycles that are available per sample from the ADC has also changed. The ADC is running at 44.1 kHz and the time between sample is given by t_{samp} .

$$t_{samp} \cdot f_{clk} = \#OfClocksPerSample = \begin{cases} 1133 & f_{clk} = 50\text{MHz} \\ 944 & f_{clk} = 41,67\text{MHz} \end{cases} \quad (5.5)$$

The new system clock frequency means that less cycles are available to process each sample from the ADC. It will be evaluated whether it is possible to implement the systems with the new lower clock frequency.

5.3 Processor Synthesis

In this section the two models will go through the synthesis in processor layer that was discussed in section 5.1. As stated earlier the goal is to have as low energy consumption as possible with hardware that is available in the FPGA, which has restricted the layout to 35 multipliers and 33216 LEs. In section 5.2 we clarified that a one-to-one mapping will be an optimal solution when synthesizing a low energy solution. The two models are going to be designed as an Final State Machine with Data path (FSMD) such the details for the system is known for registers and FUs.

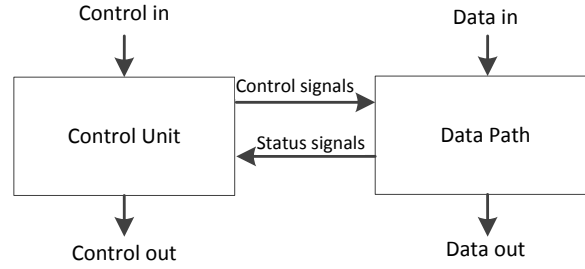


Figure 5.3: Final State Machine with Data path.

This is done in order to know all necessary details such that an estimate for energy consumption can be made. In figure 5.3 a FSMD is shown, where the data path is constructed first and the control logic to control this data path is constructed afterwards, according to the processing synthesis described above.

5.3.1 Forward-Backward Filtering Model

The first step in the synthesis process is to construct a PG. When constructing a PG of the FBF model one can easily construct a one-to-one mapping of the DFG as previously concluded. Since it requires 3 IIR filters of 4th order and each filter then needs 9 multipliers, a total of 27 multipliers would be needed in case of a one-to-one mapping. Since 35 multipliers is available this could be considered. If the model is considered further some important conclusion can be made in order to save energy. As it can be seen in figure 2.9 that shows the FBF model, one of the time reversed filtering filters is at all times receiving a zero input. A zero multiplied by any number will at all times result in a zero output from a given multiplication. Since a direct form 1 IIR filter structure has been chosen, it means that all feed-forward coefficients and the additions of these can be turned off when 5 zero inputs are received at the input to the filter that generates the tailing response. This means that only the recursive part of the DFG has to be processed from sample 6 to sample J in each block when the signal turns into a zero value signal. For a hardware shared solution a Signal-Flow Graph (SFG) often is carried out at this point in order to know the processing order. In this design process where a one-to-one mapping is conducted this part is left out and three PGs are made with knowledge from the DFG of the Direct form 1 IIR filter in figure 2.10.

In figure 2.9 in section 2.1.2, which shows the FBF model, the filters are alternately processing the zero value input signal. For this implementation it has been chosen to do otherwise. It has been chosen to use an IIR filter which process the leading response at all times and use another IIR filter which process the tailing response at all times. In order to keep an overview of the system a PG has been made for each of the filters in the model. In the PG each state needs a control signal from the control path to execute, and each state corresponds to one clock cycle. This also

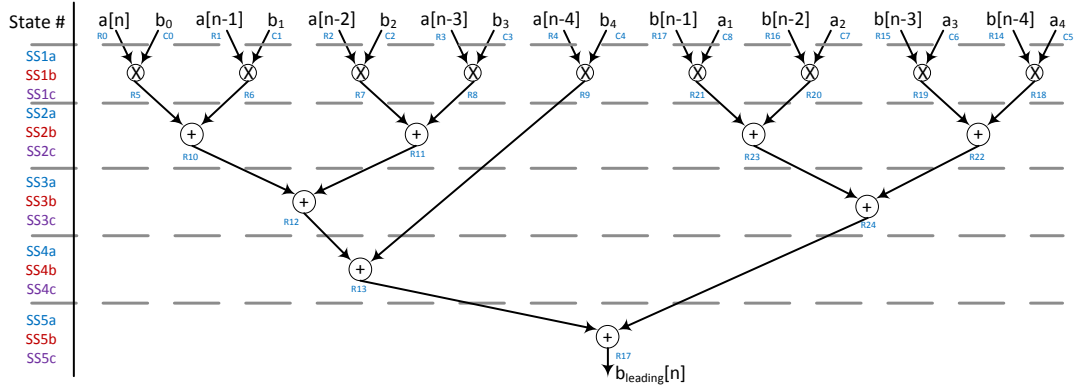


Figure 5.4: Precedence graph of Leading IIR filter, where the result needs to be saved in the 2L sample FIFO buffer. Some of the states have three names since this PG is running along side the PG in figure 5.5. No matter which of the three sub-state are active the operations in this PG are executed.

means that the used multipliers are assumed to be able to execute within 1 clock cycle. In figure 5.4 a PG for the leading filter structure is shown. The PG shows the leading filter which is a 4th order direct form 1 IIR filter. The PG is carried out in a As-Soon-As-Possible (ASAP), utilizing all inherent parallelism. This is done due to the fact that each operation will have each their dedicated FU. In figure 5.5 a PG is shown for the structure of the filter that generates the tailing response. It can be seen that four different colors now are introduced. The black marks that it is active in all scenarios. The blue shows the PG for the first 5 samples in a block length, and the red shows the PG from sample 6 to sample J in the block. The purple shows the case where the filter that generates the tailing response is off, and the leading response is fed directly to the LIFO. When organizing the processing like this the previous inputs and outputs from the filter that generates leading response needs to be shifted to

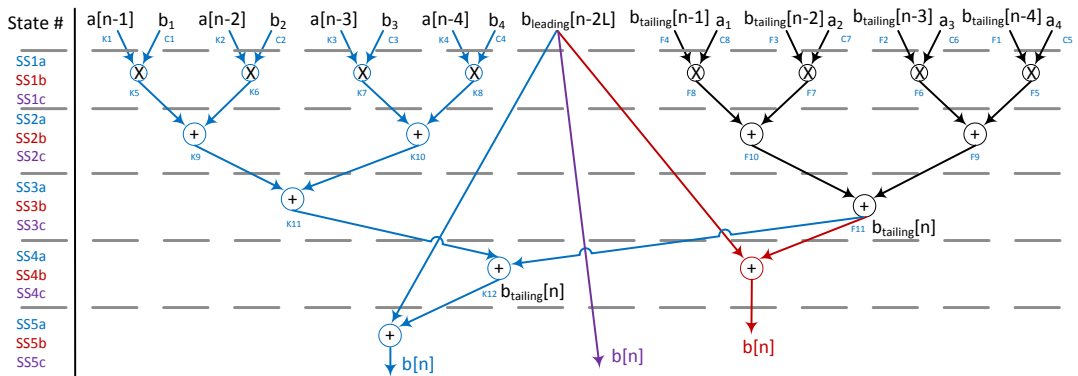


Figure 5.5: Precedence graph of tailing IIR filter including the additional adder that appear in the model. Notice that $a[n]$ will always be zero in the filter that generates the tailing response.

the filter that generates the tailing response. The last adder in both scenarios is the adder that originates from the model where the output from the buffer and output from the filter that generates the tailing response is added together. After this result $b[n]$ is computed and needs to be time reversed to generate $c[n]$ such it can be filtered in the forward filtering filter. The PG to the forward filtering IIR filter is shown in

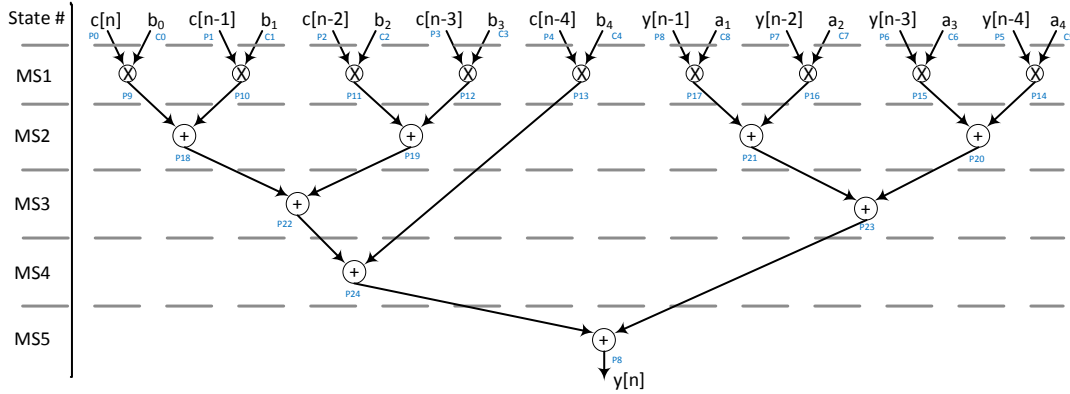


Figure 5.6: Precedence graph of Forward filtering IIR filter

figure 5.6. The PG is similar to the PG for the leading response filter but input and output values have different names. All intermediate variables have been defined in the PGs. These variables are mapped into registers with the given name. The RTL design of the data path has then been designed with the knowledge gained in the PGs. The RTL design can be seen in figure 5.7. In the figure it can be seen that the proposal to use one LIFO buffer as discussed in section 3.6 is carried out. With this approach the register shifting of used data is optimal, since all registers are used at all times. The LIFO is going to be designed as 18 bidirectional shift registers of length L . This makes possible to shift in new inputs $x[n]$, but also shifting in results from the time reverse section. An example of two bidirectional shift registers of 3 bit, is shown in figure 5.8. This means that a 2 bit variable can be push or pulled from both left and right, depending on whether the mode input is set for shifting right or left. For this FBF model the example is extended such that a 18-bit variable can be pushed and pulled from an L long shift register. This makes it possible to use this as the time reversal block. The little example has been implemented in appendix E and it can be seen in figure E.1 that the VHSIC Hardware Description Language (VHDL) code generates the RTL structure seen in figure 5.8. For further clarification of the bidirectional shift registers the VHDL-code in appendix E describe the functionality as well.

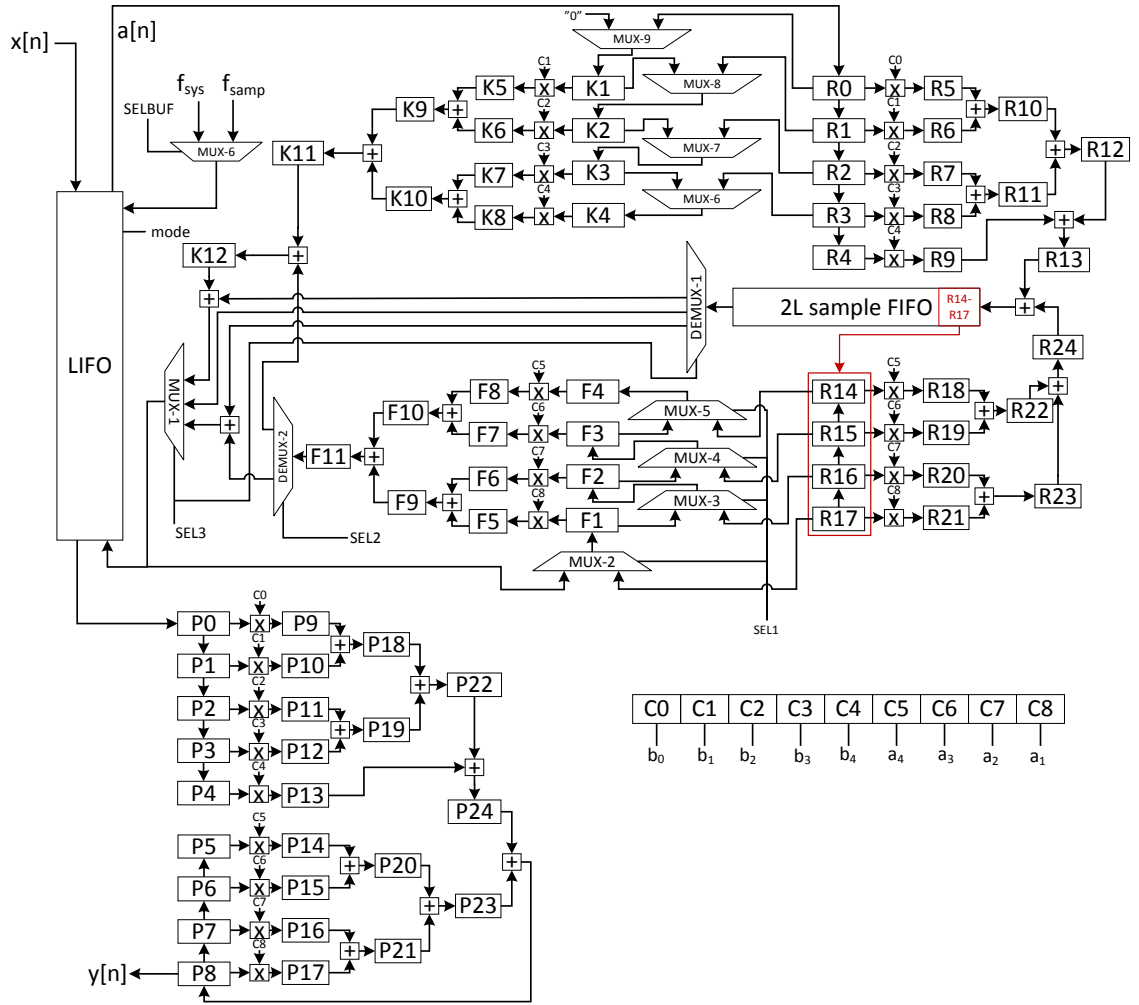


Figure 5.7: RTL level structure of the data path to the FBF model.

Control Path

The purpose of the control path is to execute the registers and control the selector signals to the multiplexers. In the synthesis process of the control path for this FBF model it has been chosen to design a conceptual state machine for clarification purpose, before going in to the details. It is important to keep things as simple as possible in order to keep the overview. The conceptual state machine can be seen in figure 5.9. Having this state diagram, it is possible to define when each of the states from the PGs of the FBF model has to be executed. In figure 5.10 a Moore machine has been defined. The main state machine is shown to the left while the sub-states are shown to the right which will be executed if the final sample of a block length is

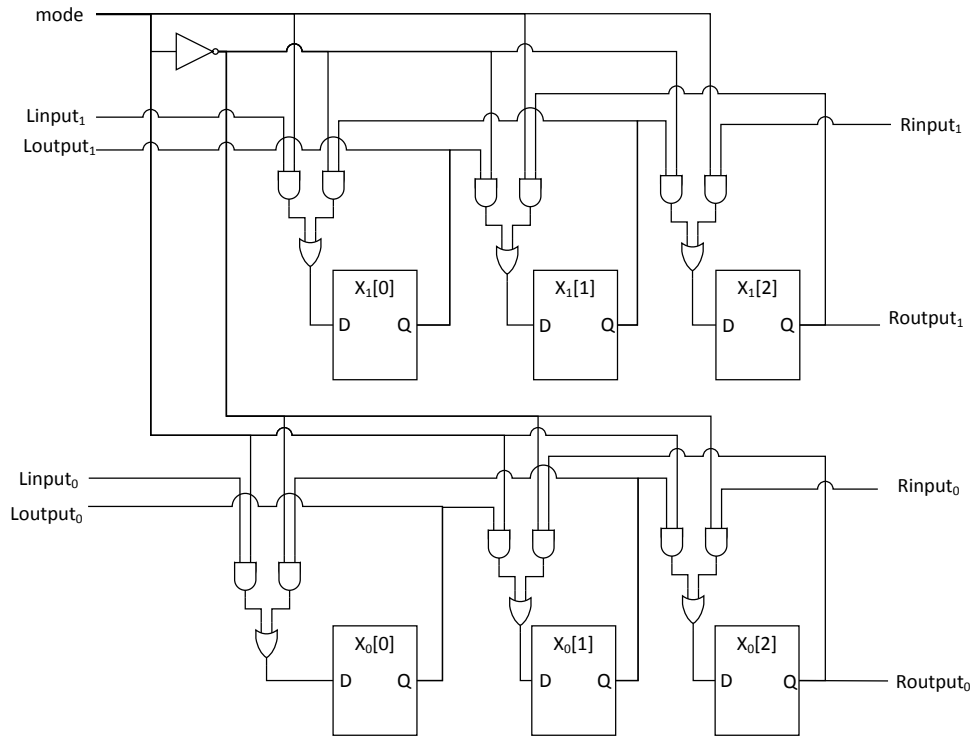


Figure 5.8: Example of a bidirectional shift register with a length of 3 bit and in parallel such that a 2 bit variable can be push or pulled from both left and right. Clear and clock signals has been left out in order simplify the figure.

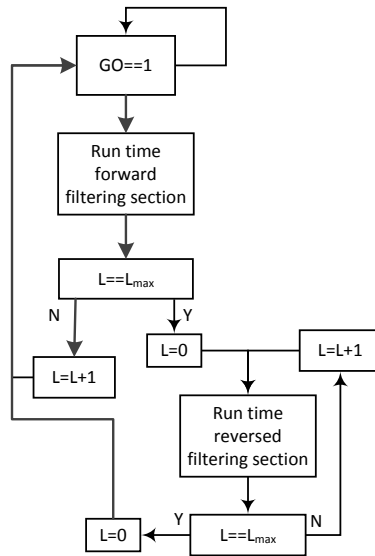


Figure 5.9: Conceptual state machine for the FBF model.

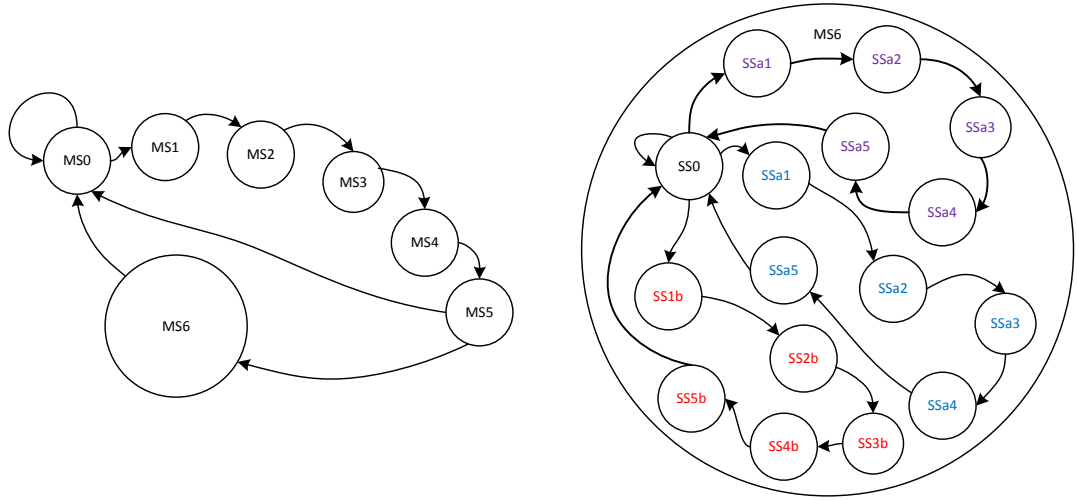


Figure 5.10: State machine for the FBF model with reference to the individual states in the PGs. Left state machine is the Main State (MS) machine, and the right state machine shows the Sub-State (SS) in MS6.

reached. The most energy efficient filter with linear phase is the wanted out come. Therefore, taking advantage of the block length considerations in section 3.6 it is now possible to derive how long it is possible to design the block length. From above it is known that 944 clock cycles are available per input sample. The first 5 cycles are used by the first 5 main states, leaving 939 cycles for the time reversed section. From the PGs it is seen that 5 states or clock cycles is required to compute one sample in the time reversed section. This means that we can derive the maximum block length for the architecture.

$$L = \left\lfloor \frac{\#available_cycles}{\#cycles/sample} \right\rfloor = \left\lfloor \frac{939}{5} \right\rfloor = 187 \quad (5.6)$$

The maximum block length L is then found to be 187 samples. In section 4.2.1 where the fixed point filter coefficients were designed it was found that the minimum block length is found to be 97 samples. This length is then defined as $J=97$ samples, and the control unit then need to be defined. A regular program counter could have been used to design the control unit but this would require a state register for each state in the L length block and suitable sub-states. Therefore another logic structure has been developed, that can describe each step in the model by its state registers. In figure 5.11 a next state logic and state logic are combined. The L -bit shift register is a simpler counter which shifts in 1 bit from the left side every time the clock pin is active. When this shift register is filled with ones, the register is reset, and the last bit in the register is called L_{max} . Main state 1 (MS1) is activated when f_{samp} shifts to high and the system clock rises. f_{samp} is the "ready" signal from the input device, such as a ADC. This one high bit will shift through the main state register until it reaches MS5. The output of the main state shift register (MS5) is fed to an

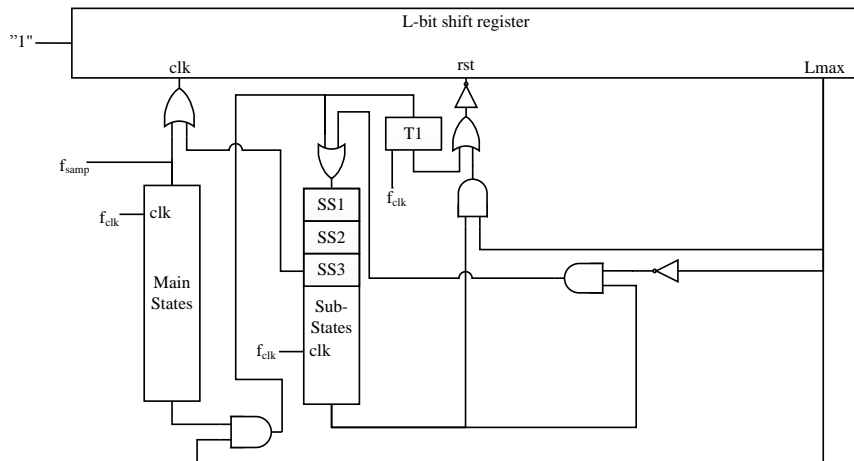


Figure 5.11: Control logic and state registers for the control unit to FBF model.

AND-gate together with L_{max} . If both are logic high the sub-state machine will then start circle in the sub-states until L_{max} is logic high again. A design of output logic that delivers control signals can now be defined on the basis of these state registers. It is not possible to define the amount of used logic in the tool used to estimate the energy consumption the additional logic is not defined further.

5.3.2 Reference Filter

The methods that are used in the synthesis process of reference filter are the same that were used to carry out the data path for the FBF model. As it was seen in section 2.2 the reference filter is a folded FIR filter. In section 4.3.1 it was found that a 89th order FIR filter would fit the specification. The PG and the RTL level

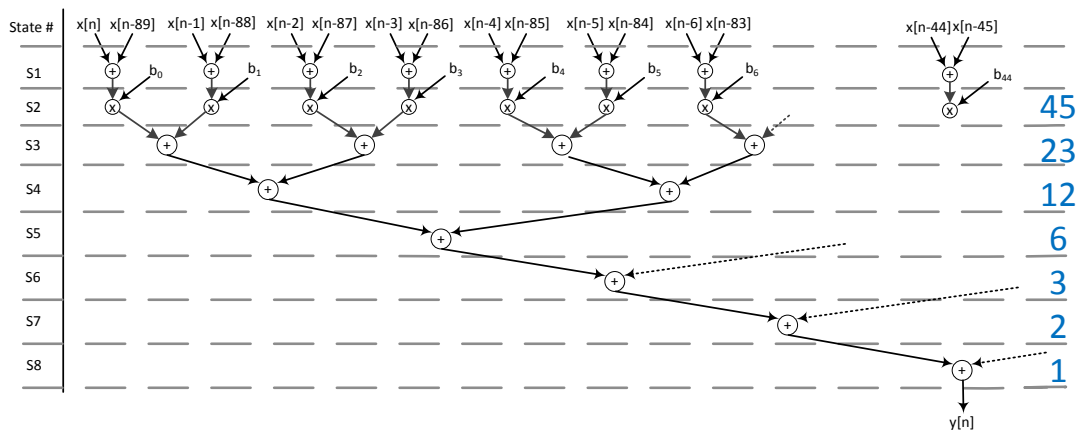


Figure 5.12: Part of precedence graph for a 89th order folded FIR filter, with ASAP scheduling. The blue number states how many variables there are needed to be added together.

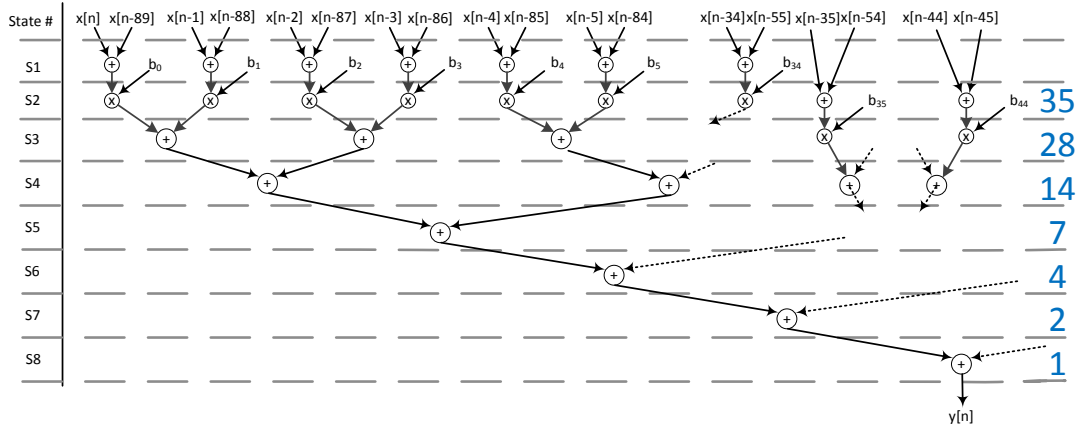


Figure 5.13: Example of precedence graph for a 89th order folded FIR filter. The blue number states how many variables there are needed to be added together.

diagram will be big. Therefore the figure will only illustrate some of the actual PG and the RTL is showing a smaller example that shows the important considerations. In figure 5.12 the PG of the folded FIR filter is shown partly. It is important to notice that all multiplications are in state number 2. The blue number shows the remaining variables that needs to get added together. The PG is given by scheduling in ASAP but this is not possible since only 35 multipliers are available and 45 are required to do so. Therefore 10 multiplications are needed to be scheduled to state number 2. In figure 5.13 the PG that shows this scenario is illustrated. Even though

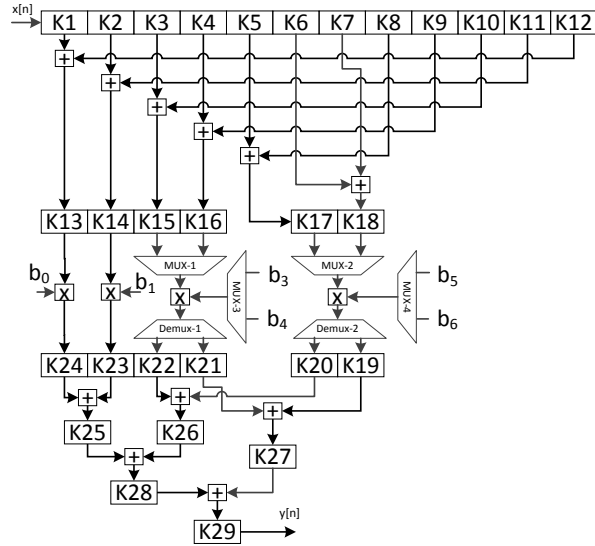


Figure 5.14: Example of the RTL level diagram for the folded FIR filter structure. This example is showing an 11th order folded FIR filter.

10 multiplications is delayed by one clock cycle the operations finishes within the same number of states. The variable names have not been allocated on the arc in the figure in order to map them to an RTL level since it is not possible to show the whole illustration. Instead this is illustrated by an example with 4 multipliers for a 11th order folded FIR filter. In figure 5.14 the example of filter is shown on the RTL abstraction level. It can be seen that each of the shared multipliers needs 2 multiplexers and a demultiplexer. This example is extended for the actual RTL design.

Control Path

For the FIR filter the control path is a more straight forward. The 8 states have to run once every time a new sample arrives. Due to the high number of parameters it is here chosen to make a conceptual state machine describing the states. The conceptual state machine can be seen in figure 5.15, and the regular state machine in figure 5.16. This state machine is made as one single logic shift register where the logic high value from f_{samp} is shifted in. This high bit will then shift through the

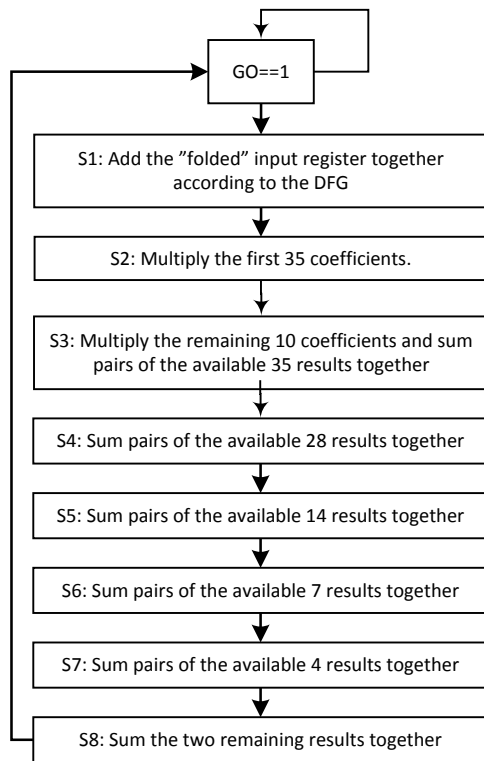


Figure 5.15: Conceptual state machine for the folded FIR filter.

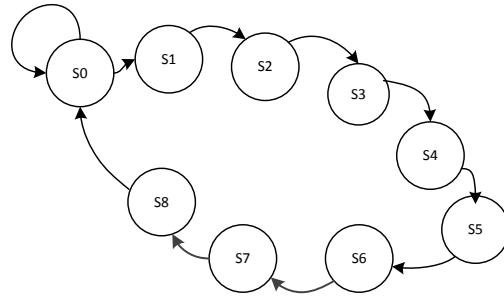


Figure 5.16: State machine for the folded FIR filter.

8-bit shift register. The shift register can be seen in figure 5.17.

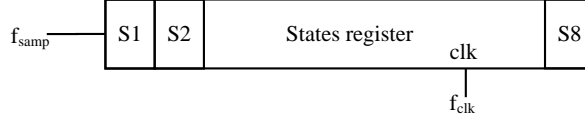


Figure 5.17: State register for the folded FIR filter.

5.4 Energy Consumption Estimation

In this section an power consumption estimate of the two models is carried out. This is done by using Altera PowerPlay Early Power Estimator [3]. This is a tool where it is possible to specify the used LEs and the controls signals to a given design. The frequency these LEs is executed at is specified together with the percentage of toggle in the signal. Toggle is an expression for the randomness in the signal.

In this project it has been chosen to set the toggle to 40 % at the input signal. In the FBF model the signal is filtered twice and therefore the output of the time reversed filtering section will have a lower toggle percent. It is chosen to use 10 % toggle for the forward filtering filter. Same amount of toggle is specified for half of the bidirectional shift register (BSR). A clock signal shift logic level every sample will have a toggle percentage at 100 %. The input sample rate was specified above to be 44.1 kHz. In table 5.1 all parameters to the Altera PowerPlay Early Power Estimator are listed. The result in the table is specifying the number of events per sample. E.g. the filter that creates the leading response in the time reversed section has 9 multipliers, and these are executed every time a new sample arrives. The filter that creates the tailing response has a variation in the number of multiplier, adders, and registers since it changes according the number of sample in the block. Therefore an average per sample of these has been calculated. E.g. the number of execution of registers in the filter that generates the tailing response is calculated as below.

$$\#ofLEs = \frac{\#Executed_registers}{Block_length} \cdot Wordlength \quad (5.7)$$


$$= \frac{12 \cdot 5 + 11 \cdot J}{187} \cdot 18 = 6.02 \cdot 18 = 108.5 \rightarrow 109 \quad (5.8)$$

This means that in average 109 LEs per sample are activated in the filter that generates the tailing response.

A LE can be programmed to different modes. One of them is Arithmetic Mode and the functioning logic is shown in appendix F. A LE in Arithmetic Mode consists of a full adder and a flip-flop. But in the Altera PowerPlay Early Power Estimator it is not possible to specify whether the LE is purely used as a register or as a full adder and register. This also means no extra power consumption is added due to

Model	Type	Section	executed/sample	# of LEs	Toggle
FBF model:	Multipliers:	Leading	9		40%
		Tailing	3		40%
		Forward	9		10%
	Adders:	Leading	8		
		Tailing	3		
		Forward	8		
	Registers:	Leading	25	450	40%
		Tailing	6.02	109	40%
		Forward	25	450	10%
		BSR1	187	3366	40%
		BSR2	187	3366	10%
		2L Buffer	374	6732	10%
	Control:		12 (bit)	12	100%
Folded FIR	Multipliers:		45		40%
	Adders:		85		
	Registers:		226	4068	40%
	Control:		8 (bit)	8	100%

Table 5.1: Specification of the number of times that the units are executed per input sample. J was found to be 97 samples and the block length is set to be 187 samples.

use of adders. The multipliers are specified separately together with frequency and the amount of toggle. Looking at table 5.1 it is seen that the FBF model requires in average 19 additions per input sample while the folded FIR needs 85 additions per input sample. This might give some percentage higher to the folded FIR than the FBF model, but this it is not possible to add in the Altera PowerPlay Early Power Estimator. The Altera PowerPlay Early Power Estimator calculates average power consumption on the base of the used frequencies and therefore this will scale linearly with the energy consumption. It is possible to specify the amount of LE where only the logic are programmed to be used of an LE. In order to know this number for both systems it would require a synthesis of the system in the Altera software. During the work with the Altera PowerPlay Early Power Estimator it was discovered that registers in the FBF model led to a higher power consumption than initially thought. Therefore the power consumption is found for the FBF model i two cases as seen in table 5.2. FBF model 1 is where $L = 187$ samples and $J = 97$ samples and FBF model 2 is where $L=J=97$ samples. This reduces the number of shift registers and this saves more power than turning of the 4 remaining multipliers and attached adders, that are turned on from sample J to sample L in FBF model 1. All information from table 5.1 has been typed into the Altera PowerPlay Early Power Estimator, and the result is shown in table 5.2. The Altera PowerPlay Early Power Estimator files can be found on the appendix-CD ./Altera/Estimation.

	Folded FIR	FBF model 1	FBF model 2
Logic	0,099	0.160	0.095
Multiplier	0,050	0.015	0.018
Clock	0,001	0.001	0.001
TOTAL	0.150	0.176	0.114
Relative Percentage	100%	117.3%	74.6%

Table 5.2: Power usage from Altera PowerPlay Early Power Estimator. All results are listed in Watt.

It is now clear that it is possible to design a linear phase filter with a lower power consumption. After having derived the results in table 5.2 can be seen that the FBF model where the block length L is specified as chosen smallest impulse response J , the power consumption is lowered from 0.15 W in the Folded FIR filter to 0.114 W in the FBF model. This has saved approximately 25%. As discussed earlier this is the average power consumption and the relative saved energy consumption will be equal.

Chapter 6

Conclusion

The purpose of this project is to define a realizable filter model with linear phase that was less energy consuming than a well-known FIR filter. FIR filters with linear phase are often realized as a folded FIR filter. This was found as a suitable reference filter. This reference filter was then evaluated against the FBF model which is a block processing model with cascaded IIR filters that outputs linear phase, due to the effects of fixed-point precision. During the analysis and design of the two models, several results were achieved regarding the three objectives, which are restated in the following.

Objectives:

1. Analyzing the FBF model proposed by [11].
2. Design a realizable fixed point systems of the FBF model, and the reference filter which is a FIR filter.
3. Compare FBF model by the use of IIR filters with the reference filter to determine in which cases the FBF model consumes less energy when implemented on a fixed-point platform, and how much energy that can be saved.

Objective 1 From the FBF model proposed by [11], an analytic derivation of the model was conducted in chapter 2. The FBF model was simulated in chapter 3 and verified to have a linear phase response. Further the amplitude response of the model was found in an appropriate matter. This was conducted by calculating the DFT of the impulse response to the FBF model when the impulse was set to be the first sample in a block. Further it was found that the phase response kept a linear phase even though the block length was cut in half. By utilizing fast processing it has been found that the group delay can be lowered from four block lengths to three block lengths.

Objective 2 For both models filters in fixed-point representation were designed in chapter 4 to fit a desired design template. A 4th order IIR filter was found suitable for the FBF model, and a fixed-point filter was used in simulator that was developed. The filter order of the reference filter was found to be a 89th order FIR filter.

Objective 3 A low-pass filter test case where the transition band was made smaller and smaller was carried out in order to determine when the FBF model has a lower number of mathematical operations than the reference filter. With the information it was concluded, that as the transition band narrows the more appropriate would it be to implement the FBF model. In chapter 5 RTL design of both models were carried out. By using Altera PowerPlay Early Power Estimator tool, the implementation of the FBF model was found to be approximately 25% less energy consumption.

6.1 Future work

Future progress could be to design the FBF model even more energy efficient than shown. Since the filter that generates the leading response in the time reversed section is reset every new block length, this filter can be step-wise turned on. Likewise for the tailing filter, the filter can be step-wise turned off for the feed-forward part. This adds even more sub-states to the state machine, which also makes the output logic more complex.

In future work could also consist of programming it in VHDL code such the actual energy consumption can be measured. In such case it would be interesting to investigate the two proposed FBF models but also comparing with an implementation using Random-Access Memory (RAM) instead of shift registers.

Bibliography

- [1] Stephen Brown and Zvonko Vranesic. *Fundamentals of Digital Logic with VHDL Design*. McGraw-Hill Science, 2008. ISBN 978-0077221430. URL <http://www.amazon.com/Fundamentals-Digital-Logic-Design-CD-ROM/dp/0077221435>.
- [2] Altera Corporation. Cyclone ii device family data sheet, 2015. URL <https://www.altera.com/products/fpga/cyclone-series/cyclone-ii/support.html#Cyclone-II-Device-Handbook--All-Sections->.
- [3] Altera Corporation. Powerplay early power estimators (epe) and power analyzer, 2015. URL <https://www.altera.com/support/support-resources/devices/power/pow-powerplay.html>.
- [4] Daniel D. Gajski, Samar Abdi, Andreas Gerstlauer, and Gunar Schirner. *Embedded System Design: Modeling, Synthesis and Verification*. Springer Publishing Company, Incorporated, 1st edition, 2009. ISBN 1441905030, 9781441905031.
- [5] Terasic Inc. Altera de2 board, 2015. URL <http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=53&No=30&PartNo=2>.
- [6] Vinay Kumar and Sunil Bhooshan. Design of One-Dimensional Linear Phase Digital IIR Filters Using Orthogonal Polynomials. *ISRN Signal Processing*, 2012 (870276), 2012.
- [7] Ayuchi Kurosu, Syoichiro Miyase, Shigenori Tomiyama, and Tsuyoshi Takebe. A Technique to Truncate IIR Filter Impulse Response and Its Application to Real-Time Implementation of Linear-Phase IIR Filters. *IEEE Transactions on Signal Processing*, 51(5):1284–1292, 2003.
- [8] Vijay K. Madisetti and Douglas B. Williams. *Digital Signal Processing Handbook*. Elsevier Inc., 1999. ISBN 978-0849385728.
- [9] Mathworks Inc. *Minimum-Order Low Pass Filter Design Using Multistage Techniques*, 2014. URL <http://www.mathworks.com/help/dsp/examples/designing-low-pass-fir-filters.html?refresh=true#zmw57dd0e220>.

- [10] Alan V. Oppenheim, Ronald W. Schaffer, and John R. Buck. *Discrete-Time Signal Processing (2nd Edition)*. Prentice Hall, 1999. ISBN 0137549202.
- [11] Scott R. Powell and Paul M. Chau. A Technique for Realizing Linear Phase IIR Filters. *IEEE Transactions on Signal Processing*, 39(11):2425–2435, 1991.
- [12] K. P. Soman R. Ramanathan. A Novel Methodology for Designing Linear Phase IIR Filters. *ACEEE International Journal on Communication*, 1(1):19–23, 2010.
- [13] D. Schlichthärle. *Digital Filters: Basics and Design*. Springer Berlin Heidelberg, 2011. ISBN 9783642143243. URL <https://books.google.dk/books?id=or7WjwEACAAJ>.
- [14] Steven W. Smith. *The Scientist and Engineer's Guide to Digital Signal Process*. California Technical Publishing, San Diego, CA 92150, second edition, 1999. ISBN 0-9660176-4-1.

Appendix A

Complexity Test

This appendix will describe how the filter order increases while step-wise making the transition band more narrow for a low-pass filter design specification. In table A.1

	Low-pass filter specifications
Bandpass Ripple [dB]	1
Bandstop Attenuation [dB]	60
Passband frequency [ω_p]	0.1
Stopband frequency [ω_s]	0.11-0.8

Table A.1: Tested filter design specification with varying stopband frequency.

some randomly chosen low-pass filter design parameters are given. As it can be seen the stopband frequency (ω_s) is starting at 0.8ω and in 100 step move linearly to 0.11ω . This makes the stop band of a size between $0.11\omega - 0.8\omega$ as seen in figure A.1. The

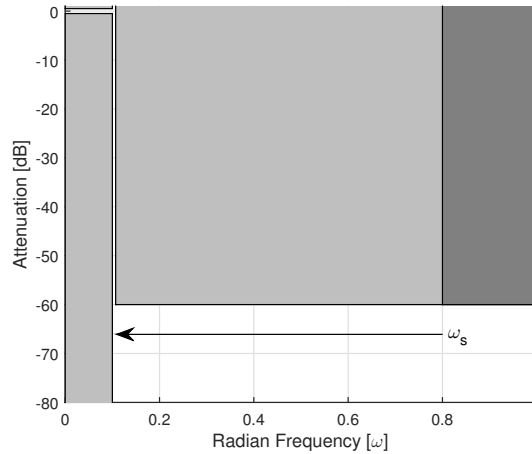


Figure A.1: Random chosen design specification from table A.1, where move ω_s in 100 steps from 0.8 to 0.11ω

reference filter and the forward-backward filtering have to comply with this design specification. In section 1.3 it is clarified that only the half gain is required in the IIR filter, and this leads to a design specifications for the IIR filter design. These two

	Low-pass filter specifications
Bandpass Ripple [dB]	0.5
Bandstop Attenuation [dB]	30
Passband frequency [ω_p]	0.1
Stopband frequency [ω_s]	0.11-0.8

Table A.2: Tested filter design specification with varying stopband frequency for the IIR filter design.

design specifications are plotted in figure A.2, and it is clear to see that the stop band has less attenuation but the ripple is more narrow for the IIR filter design. These

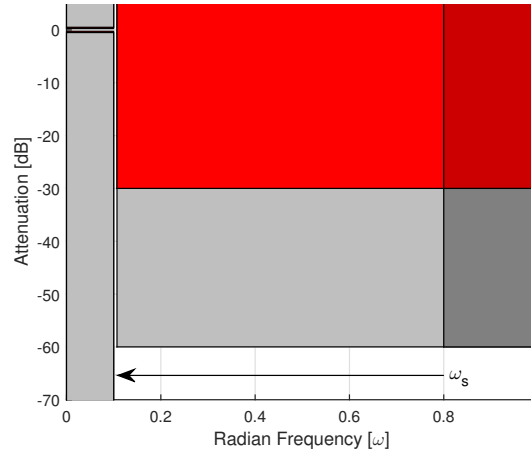



Figure A.2: Random chosen design specification from table A.1, where move ω_s in 100 steps from 0.8 to 0.11 ω

design parameters are used in a Matlab minimum order filter design function, and this will return the minimum order of the filter to the give design parameters. To determine the order for the IIR filter a function the finds the minimum order for elliptic filters is used. For the FIR filter a function which determine the optimal order of the Parks-McClellan FIR filter which is used. Since this only is a estimator 1 % more attenuation is added to the stop such that the filter will comply with the specification for all filter orders. These filter orders are then computed in equation (3.1) and (3.2) depending on whether it is the IIR or the FIR filter. The result of this is seen in figure 3.3 in section 3.1.3. The scrip for this test can be found on the appendix-CD  ./Matlab/FIRvsFBF.m

Appendix B

Simulation of Model in Matlab

In this appendix the simulation model is explained in detail.

B.1 Floating-Point Simulation

Based on the knowledge gained in section 2.1 a Matlab simulation has been designed. This Matlab simulation is based on a block processing method, instead of the sample by sample technique that a realizable model is using. The simulation in listing B.1 follows these steps:

1. In line 14 equation (2.7) is processed in to a block $a_k[n]$
2. In line 15 block $a_k[n]$ is filtered and saved in block $b_k[n]$
3. In line 16-18 all $b_k[n]$ block's is summed together 3 block's ahead compared to the input. These 3 blocks originate from the first block time reversal and the 2L sample delay from equation (2.11).
4. In line 21 $b[n]$ is time reversed into $c[n]$ according to equation (2.8) plus a L sample delay.
5. Finally a standard filtering process is performed in line 23 on the output $c[n]$ to acheive the output $y[n]$

The Matlab simulation is seen below in listing B.1.

```
1 function [ Y ] = lpiirfilt(X, a, b, B1)
2 % lpiirfilt Summary of this function goes here
3 % Inputparameters:
4 % X = input
5 % a, b = coefficients for IIR filter
6 % B1 = Block length
7
8 N = length(X);
```

```

9  Xa = zeros(floor(N/B1)+1,B1+B1);      %a[n]
10 Xb = zeros(1,N+B1);                  %b[n]
11 Xc = zeros(1,length(X+2*B1));        %c[n]
12 Y = zeros(1,length(X+2*B1));        %y[n]
13 for ii = 1:floor(N/B1-1)
14     Xa(ii,:) = [flip(X((ii-1)*B1+1:ii*B1)) zeros(1,B1)]; % LIFO
15     Temp(ii,:) = filter(b,a,Xa(ii,:)); % filter with tail
16     Xb((ii+1)*B1+1:(ii+2)*B1) = Temp(ii,B1+1:end)+Xb((ii+1)*B1+1:...
17     (ii+2)*B1); % add leading response to previous tail
18     Xb((ii+2)*B1+1:(ii+3)*B1) = Temp(ii,1:B1); % save new tail
19 end;
20 for ii = 1:floor(length(X)/B1-2)
21     Xc(ii*B1+1:(ii+1)*B1) = flip(Xb((ii-1)*B1+1:ii*B1)); % LIFO
22 end;
23 Y = filter(b,a,Xc); % forward filtering IIR filter
24 end;

```

Listing B.1: Info of fixed-point object in Matlab.

The simulation model is tested in section 3.2.1.

B.2 Fixed-Point Simulation

The model in the previous section is changed into a fixed point simulator by using Matlabs "Fixed-Point Designer" package. All intermediate parameters X, Xa, Xb, Xc, Xd and Y are changed to fixed point objects and the filters are specified to use fixed-point arithmetic where it is possible to decide precision of variables, coefficient, input/output, multiplier and accumulator. It is a method to secure that all variables are described by the correct precision and quantized in the intended manner. The fixed-point simulation model can be found on the appendix-CD

📁./Matlab/lpiirfiltQuanXbit.m

Appendix C

Design of Filters using Matlab

In this appendix the filters for the FBF model and the reference filter are designed in Matlabs fixed-point designer. The tool is used since it gives a great overview of the ongoing design of the fixed-point filter. The restriction from the template in section 4.1 is defined as following for the Matlab script.

```
1 rp = 0.50;          % Passband ripple
2 rs = 60;            % Stopband attenuation
3 Fs = 2;             % Sampling frequency
4 f = [0.1 0.15];     % Cutoff frequencies
5 a = [1 0];          % Desired amplitudes
```

These parameters are used in the design functions that design the filters for the FBF model and the reference filter.

C.1 Fixed-Point Filter for the FBF Model

To design an elliptic IIR filter it is needed to determine the minimum filter order.

```
1 [ellip_n,Wn] = ellipord(f(1), f(2), rp/2, rs/2);
```

The filter order is used to derive the filter coefficients.

```
1 [ellip_b,ellip_a] = ellip(ellip_n, rp/2, rs/2, f(1));
```

The filter coefficients are then applied in a direct form 1 IIR filter object.

```
1 Hdf1 = dfilt.df1(ellip_b,ellip_a)
```

Hdf1 is now a filter design object that can be used to filter the signals. Before turning the coefficients into fixed-point precision we multiply the scale factor to the feed-forward coefficient or numerator as matlab names these coefficients.

```
1 s = floor(sqrt(1/sum(abs(filter(ellip_b,ellip_a, ...
2     [1 zeros(1,5000)]).^2))*2^17)/2^17
3 Hdf1.Numerator = s*ellip_b
```

The filter object is then changed to use fixed-point arithmetic by using following command

```
1 Hdf1.Arithmetic = 'fixed';
```

After this parameter is set it is possible to specify the precision on each of the internal variables of the filter. Since we want to use the 18-bit Multipliers it is chosen to use these 18-bit for the FBF model, and a parameter called *wordlength* = 18 is defined to specify this. As it can be seen in some of the denominator values (feedback

Numerator:	Denominator:
0.0941	1
-0.2735	-3.4334
0.3772	4.5382
-0.2735	-2.7249
0.0941	0.6263

Table C.1: Coefficients for the IIR filter to the FBF Model.

coefficients) is larger than 1. This means that the dynamic range of the coefficients needs to have at least 3 bits to describe the integer value. In Matlab we need to specify the word length including the sign bit and we need to define how many bits are used to describe the fraction. Since we have chosen to use 18 bit word length, the feedback coefficients need 1 bit for the sign value and 3 bits for the integer which leaves 14 bits for the fraction part. All other values are defined as 18 bit word length and 17 bit fraction. The rounding mode is defined as "*floor*" meaning that it uses truncation. The overflow mode is specified as wrap, which means that the signal values can "*Wrap – Around*" the dynamic range. This also means that the only place that overflow should be considered is on the output for the Direct form 1 IIR filter. This means that when we have specified all internal variables we will see the information of the fixed-point filter object in Matlab as in listing C.1.

```
1 Discrete-Time IIR Filter (real)
2 -----
3 Filter Structure      : Direct-Form I
4 Numerator Length     : 5
5 Denominator Length   : 5
6 Stable               : Yes
7 Linear Phase         : No
8 Arithmetic           : fixed
9 Numerator            : s18,17 -> [-1 1)
10 Denominator          : s18,14 -> [-8 8)
11 Input               : s18,17 -> [-1 1)
12 Output              : s18,17 -> [-1 1)
13 Numerator Prod       : s36,35 -> [-1 1)
```

```

14 Denominator Prod      : s36,35 -> [-1 1)
15 Numerator Accum       : s36,35 -> [-1 1)
16 Denominator Accum     : s36,35 -> [-1 1)
17 Round Mode            : floor
18 Overflow Mode         : wrap
19 Cast Before Sum       : true

```

Listing C.1: Info of fixed-point IIR filter object in Matlab.

This filter object can then be used in the simulator of the FBF model that was described in appendix B.

C.2 Reference Filter

Almost the same procedure is used on the reference filter. For the simulation purpose the filter is not implemented as a folded FIR filter, since this makes no change to the results. First we determine the needed order of the Parks-McClellan FIR filter, as it is done in listing C.2 line 1 and 2. On line 3 the coefficients are found and in line 4 they are stored in the filter object *Hdfir*.

```

1 dev_pm = [(10^(rp/20)-1)/(10^(rp/20)+1)  10^(-(rs*1.01)/20)];
2 [N_fir_pm, fo, ao, w] = firpmord(f,a,dev_pm,Fs);
3 FIR_b_PM = firpm(N_fir_pm, fo, ao, w);
4 Hdfir = dfilt.dfir(FIR_b_PM)

```

Listing C.2: FIR filter object

We specify the over flow and round mode as we did it earlier on the IIR filter. All internal variables are defined as we did earlier as well, by having on sign bit and one minus word size as the fraction. The fixed-point filter object information in Matlab is shown in listing C.3.

```

1 Discrete-Time FIR Filter (real)
2 -----
3 Filter Structure   : Direct-Form FIR
4 Filter Length     : 89
5 Stable            : Yes
6 Linear Phase      : Yes (Type 1)
7 Arithmetic        : fixed
8 Numerator         : s18,17 -> [-1 1)
9 Input             : s18,17 -> [-1 1)
10 Filter Internals  : Specify Precision
11   Output          : s18,17 -> [-1 1)
12   Product         : s36,35 -> [-1 1)
13   Accumulator     : s36,35 -> [-1 1)
14   Round Mode      : floor
15   Overflow Mode   : wrap

```

Listing C.3: Info of fixed-point FIR filter object in Matlab.

Appendix D

Impulse Response and Block Length Considerations

The Argument that is the ground stone of the FBF model is the theory of that an IIR filter implemented on a fixed-point platform has a finite impulse response. The impulse response of the IIR filter that was designed for the FBF model in section 4.2.1 is shown here. Figure D.1 shows the impulse response when the filter is evaluated

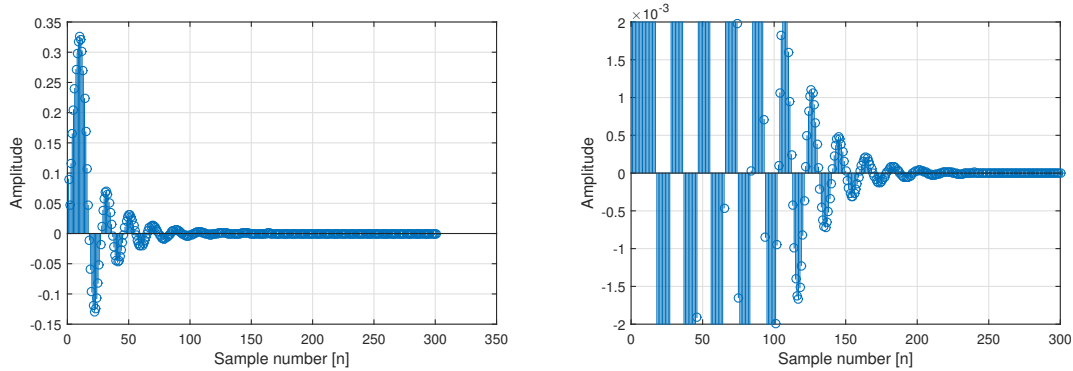


Figure D.1: Impulse response of IIR filter when evaluated on a floating point system.

in floating point and figure D.2 shows the impulse response when it is evaluated in fixed-point arithmetic. For the floating-point case the impulse response goes against zero, but the the fixed-point thing look more different. It can be seen that it enters a zero input oscillation after approximately 150 samples. Another concern could be if the variance of the input is too small, the time reversed filtering might contribute with zero input limit cycle oscillation to the output contentiously.

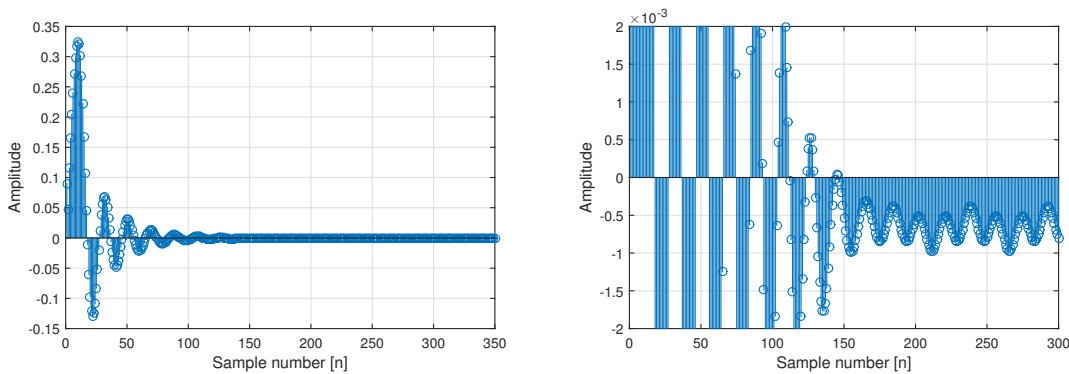



Figure D.2: Impulse response of IIR filter when evaluated on a fixed-point system.

Appendix E

Implementation of bidirectional Shift Register in VHDL

In VHDL the understanding of the bidirectional shift registers are much simpler to describe. One could write VHDL code that described the exact logic that was shown in figure 5.8, and get the correct design, but even simpler VHDL code is possible to write. A small VHDL program has been written and are available on the appendix-CD ./BSR/BidirectShift.vhd and the important lines are here in Listing E.1 shown, and the resulting RTL level figure generated in Altera Quartus II shown in E.1.

```
1  elsif (rising_edge(CLOCK_50) and mode = '1') then
2      R_out(0) <= temp0(0);
3      R_out(1) <= temp1(0);
4      temp0((L-1) DOWNT0 1) <= temp0((L-2) DOWNT0 0);
5      temp1((L-1) DOWNT0 1) <= temp1((L-2) DOWNT0 0);
6      temp0(0) <= L_in(0);
7      temp1(0) <= L_in(1);
8  elsif (rising_edge(CLOCK_50) and mode = '0') then
9      L_out(0) <= temp0(L-1);
10     L_out(1) <= temp1(L-1);
11     temp0((L-2) DOWNT0 0) <= temp0((L-1) DOWNT0 1);
12     temp1((L-2) DOWNT0 0) <= temp1((L-1) DOWNT0 1);
13     temp0(L-1) <= R_in(0);
14     temp1(L-1) <= R_in(1);
15 end if;
```

Listing E.1: BSR.

In Listing E.1 L describes the block length, L_in/L_out describes the left I/O, and R_in/R_out describes the right I/O.

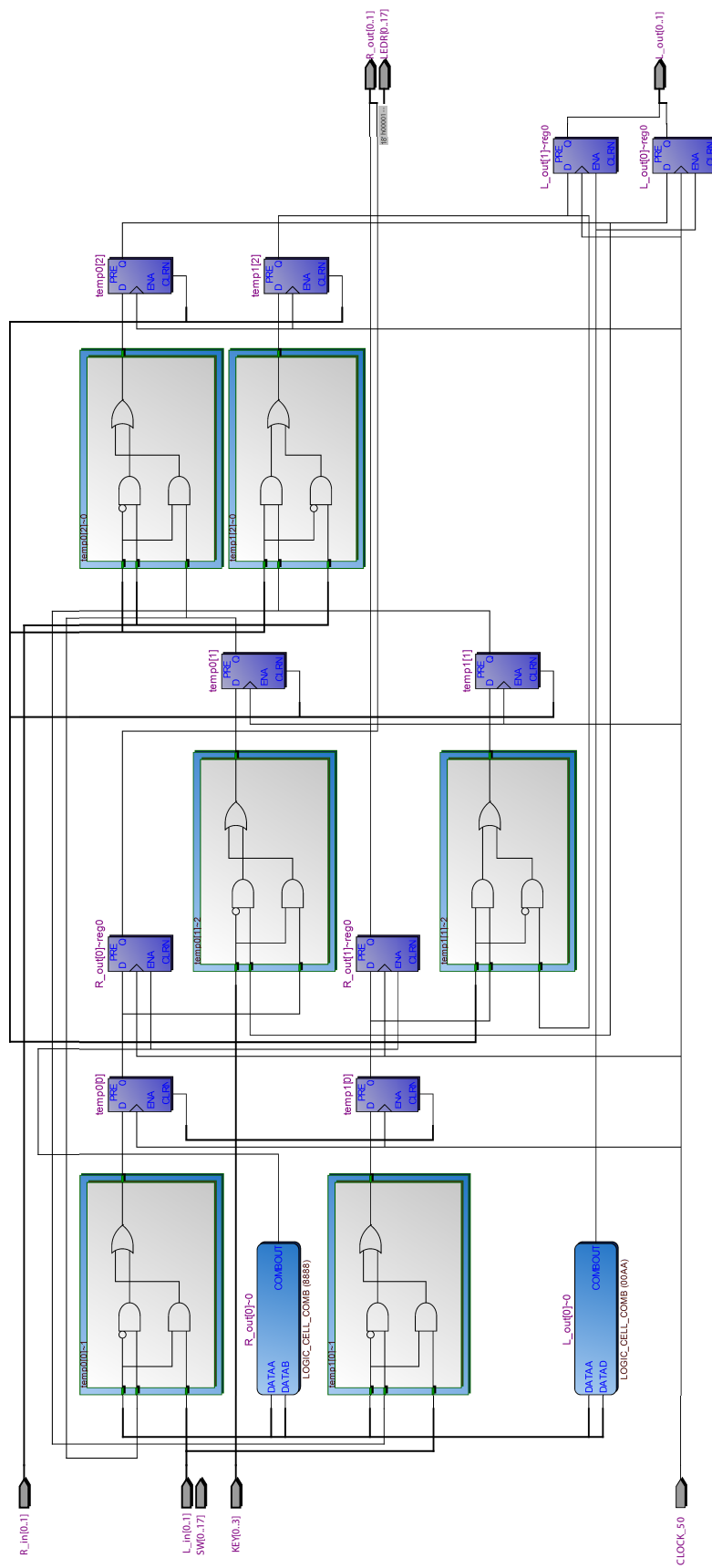


Figure E.1: RTL level figure generated in Altera Quartus II, with the VHDL code designed in this chapter.

Appendix F

Logic Elements

An Altera FPGA consists of LEs where the structures change between the device families. LEs typically consist of a flip-flop and some logic, that can be programmed to different functionalities. In an Altera Cyclone II Architecture the LE can be programmed to different modes.[2] One mode is called Arithmetic Mode and the mode is shown in figure F.1. This mode programs the logic to be a full adder. This means the adder is a part of the logic element that also contains a flip as seen in figure F.1.

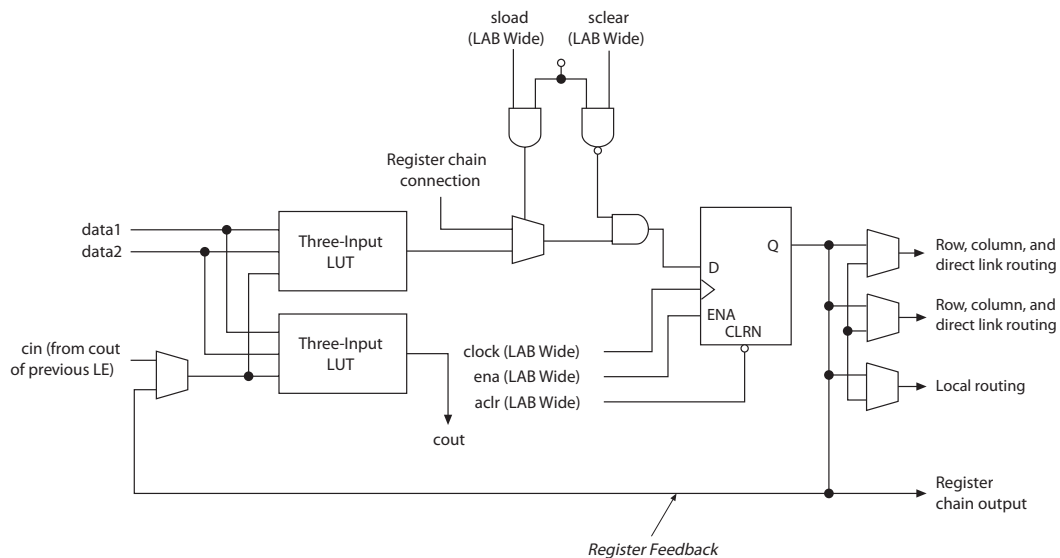


Figure F.1: logic element of an Altera Cyclone II FPGA in Arithmetic Mode. [2]