ECOLE NATIONALE SUPÉRIEURE DE
L'ELECTRONIQUE ET DE SES APPLICATIONS

END-OF-STUDY PROJECT

Real-Time Sample Rate Converter for High-Quality Audio Signal Processing

Author:
Gabriel SUC

Supervisors:
Dr. Pablo MARTÍNEZ-NUEVO
Dr. Sven Ewan SHEPSTONE
Mr. Henrik Lund JENSEN
Dr. Geoff MARTIN

Academic Supervisor:
Dr. Matthieu GUERQUIN-KERN

A report submitted in fulfillment of the requirements for the diploma of Electrical Engineer

in the master's program of Multimedia Systems

April 1, 2018 - September 30, 2018

ECOLE NATIONALE SUPÉRIEURE DE L'ELECTRONIQUE ET DE SES APPLICATIONS

Preface

E.N.S.E.A.
Multimedia Systems

Electrical Engineer

Real-Time Sample Rate Converter for High-Quality Audio Signal Processing

by Gabriel SUC

This report has been made by Gabriel Suc, a student in the master's program of Multimedia Systems at Ecole Nationale Supérieure de l'Electronique et de ses Applications.

The report details the end-of-study project carried out at Bang & Olufsen in Struer, Denmark between the months of April and September of 2018.

The content of this report is confidential due to the information regarding the company and the content of the project itself.

Acknowledgements

I would like to thank warmly Dr. Pablo Martínez-Nuevo and Dr. Sven Ewan Shepstone for giving him the opportunity to undertake this project within Bang & Olufsen. I would also like to thank them for their help, advice and support throughout the project as well as Prof. Matthieu Guerquin-Kern.

Moreover, the help of Mr. Henrik Lund Jensen and Dr. Geoff Martin was precious and is truly appreciated.

Finally, I would like to say thank you to the R&D Acoustics departement for their welcome and great spirit, especially Prof. Søren Bech and Mr. Lars Jørgensen.

Contents

List of Figures

List of Tables

List of Abbreviations

FIR	**F**inite **I**mpulse **R**esponse
IIR	**I**nfinite **I**mpulse **R**esponse
DSP	**D**igital **S**ignal **P**rocessing
A/D	**A**nalog-to-**D**igital
D/A	**D**igital-to-**A**nalog
DTFT	**D**iscrete-**T**ime **F**ourier **T**ransform
IR	**I**mpulse **R**esponse
SRC	**S**ample **R**ate **C**onversion
MPOS	**M**ultiplications **P**er **O**utput **S**ample
LPF	**L**ow**P**ass **F**ilter
dBFS	**D**ecibels relative to **F**ull **S**cale
WAV	**Wav**eform Audio File Format
SNR	**S**ignal to **N**oise **R**atio

List of Symbols

Ω	Analog Frequency
Ω_N	Nyquist Frequency
ω	Digital Frequency
dB	Decibel

Chapter 1

Introduction

Most of the modern signal processing chains consist of an interconection of digital discrete-time systems operating at different rates. For instance, let us consider a guitar player who desires to record his music. Thanks to microphones, the signal that the player is producing will be digitalized and will go inside the digital signal processing chain. From recording to reproduction, a variety of systems are involved and they operate at different rates. So as to fully reconstruct the signal, all the systems must perform at the same rate. As a result, sampling rate conversion is crucial in order to make these interconnections possible. Moreover, for the sake of real-time processing, efficiency is one of the major aspect of this topic. Essentially, a sample-rate converter would be considered efficient if the data is processed fast and the signal obtained is faithful to the original.

First and foremost, we can distinguish two categories of conversions: synchronous and asynchronous. When a system has only one single internal clock, i.e. when the system has a constant output sampling rate, we will use a synchronous converter (for example, we can think about a system accepting the following set of input sampling rates: $F_{s_{in}} = \{32\,\text{kHz},\ 44.1\,\text{kHz},\ 48\,\text{kHz}\}$ with as only output sampling rate: $F_{s_{out}} = \{48\,\text{kHz}\}$). On the other hand, when a system has multiple clocks, we will choose an asynchronous sample rate converter (in this case, we can imagine a USB device receving data from a PC and converting them to an analog signal, operating at two different rates. Another example would be when two systems are said to have the same clock, but they slightly differ over time).

The purpose of this report is to investigate and document different methods to create a real-time synchronous converter that can be implemented in the platforms of Bang & Olufsen. It is an important project for the company since the current converters are responsible of audible artifacts that do not meet the quality standards of B&O. Moreover, available solutions cannot be fully applied within this context. Indeed, they are either not fully adapted to the cases required by the firm or they are suboptimal.

The idea was to use either a FIR or IIR filter implemented in a multi-stage separation and exploring the polyphase decomposition of the non-recursive part.

The report is composed of two sections:

- Bang & Olufsen. This first part presents an overview of the company and the context in which the internship took place.
- End-of-study Project. This second part describes the work done by the student during the internship in a theortical approach first and in a practical case thereafter.

Chapter 2

Work Environment

Introduction

This chapter presents in general terms the work environment and the goals of the student's end-of-study project.

It will start by a presentation of the host organisation, Bang & Olufsen, then the department where the student worked and finally expound a short expolation brief on the intership itself.

2.1 Bang & Olufsen

2.1.1 The Company

Founded in 1925 by Peter Bang and Svend Olufsen in Struer, Denmark, the company initially started by selling radio devices plugged to an outlet and thus functioning with alternating current. Compared to the other kind of equipments at that time, that were working on battery and therefore expensive and impratical, their inventions became a life-changer for a lot of people and were the beginings of their success.

Nowadays, Bang & Olufsen is well-known worldwide as a luxury brand for its distinctive design, sound quality and innovative technology of its products. The current portfolio includes:

- Wireless speaker systems (Beosound Shape, Beosound 2, ...).
- Televisions (Beovision Eclipse, Beovision Avant, ...).
- Speakers (Beolab Collection, ...).
- Portable bluetooth speakers (Beoplay A1, Beolit 17, ...).
- Sound systems (Beosound Core).

The company headquarters are still in Struer, where part of the development and the production also takes place. The other production facility is located in the Czech Republic, mainly focused on assembly and quality testing. The company site in Copenhagen deals with global sales and the B&O PLAY brand section as well as some software development. Recently, Bang & Olufsen sold its automotive business to HARMAN, but it collaborates with Hewlett-Packard to bring the B&O sound to HP's products, and it also has a recent partnership with LG Electronics on TVs and audio solutions for smartphones.

2.1.2 R&D Acoustics

The student worked in the research department, which is under the Platform Development, Research, Maintenance section. The responsibilities of this department are:

- Establish, implement and maintain research strategy to support product roadmap.
- Internal communication and alignment with key stakeholders.
- Write, coordinate and track funding applications.
- Initiate and support patent process.
- Establish and maintain collaboration with external partners (universities, companies).
- Represent B&O in relevant funding bodies and committees.

In the figure below, the current (during the time of the internship) organisation of the Acoustics & Research department is shown. It is divided into five parts: Research, Sound Quality & Design, Electro Acoustics, Acoustic Test & Validation, and Acoustic DSP.

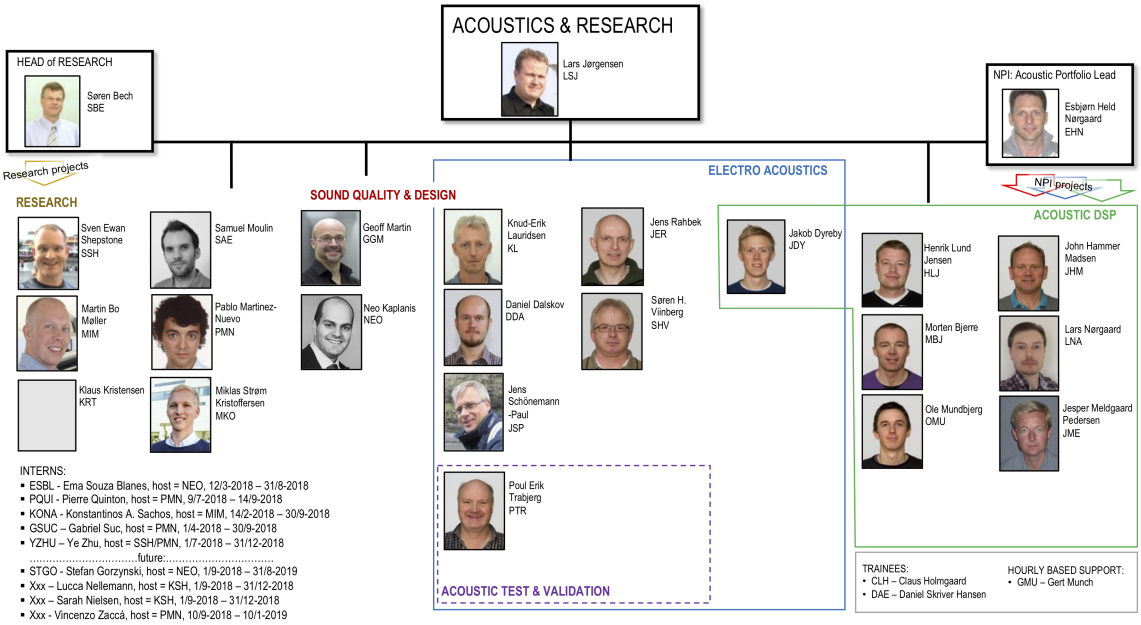


FIGURE 2.1: Organization Chart of the Acoustics & Research Department

The internship was mainly supervised by Dr. Pablo Martínez-Nuevo, providing precious help and directives on the design and implementation of the DSP algorithm. Dr. Sven Ewan Shepston gave essential input on the matter and assisted the student on the software implementation facet. Finally, Mr. Henrik Lund Jensen and Mr. Geoff Martin lent their support for the design and the implementation of the algorithm as well as establishing a performance evaluation and a set of requirements.

2.2 Exploration Brief

This section is here to define properly what the project was about and the context in which it happened. Moreover, a time structure will also be provided.

The goal of the project was to design and implement a sample rate converter that meets the quality and computational demands of a real-time audio signal processing chain and that can be integrated in future and current software platforms. Modern audio/visual devices consist of several sub-systems operating at different rates. Sampling rate conversion is the key operation that makes these blocks working together. Despite being a crucial, and widely used operation of many discrete-time processing systems, it has been broadly overlooked, often resulting in poor performance.

On one hand, open-source software implementations of sampling rate conversion used in Bang & Olufsen products (e.g. GStreamer) create audible artifacts that do not meet the performance standards of B&O audio quality. On the other hand, third-party proprietary implementations (e.g. Audio-Weaver) do not provide full support for the variety of sampling rates that are required in B&O systems. Moreover, since there is no control over its design and implementation, it is not viable to reliably evaluate and accommodate its performances to the company's needs.

Therefore, this project aims at establishing the design criteria of a sampling rate converter as well as providing a computationally efficient implementation that meets the appropriate quality demands.

The project had to explore certain points that, with the different constraints and limitations, served as guidelines:

- The impact on audio quality of the implementation of sample-rate conversion systems in GStreamer that are currently used by B&O products.
- Review the design criteria and implementation of available open-source sample rate converters.
- Establish the criteria for an appropriate performance in terms of filter design taking into account perceptual aspects.
- Explore the different tradeoffs for implementation that may involve throughput, latency, real-time operation, CPU and memory usage, or power consumption.
- Evaluate the performance of different approaches to perform linear convolution in an efficient manner.
- Design the sample-rate converter considering the knowledge gathered from the previous stages and considering the synchronous and asynchronous approaches.
- Implement the algorithm in C language and test the performances in different hardware platforms, for instance, ARM-based processors.

However, the student and the supervisors agreed that synchronous sampling rate conversion should be the principal focus for this project, knowing that the asynchronous part was already in the center of other reasearch projects.

Subsequently, the project team had to set a desired planning in order to fulfill the requirements specified above. The inial schedule was as follows, with some modifications that happened along the way:

- **1st month:** Study of sample-rate conversion systems and current approaches for implementation.
- **2nd month:** Formulate a set of requirements for the design of a sample-rate conversion system tailored to B&O's needs.
- **3rd — 4th month:** Implement a C prototype and benchmark.
- **5th month:** Turn the implementation into a GStreamer plugin.
- **6th month:** Wrap-up and conclusion.

Conclusion

As described in this chapter, we are from this point forward aware of the framework of the project. The next part will explain in detail the theoretical background needed to fully grasp the topic of the interpship.

Chapter 3

End-of-Study Project

3.1 Theoretical Part

3.1.1 Introduction

A recurrent concern has arisen with the emergence of new digital signal processing applications. In many cases, these systems are the result of an interconnection of distinct devices operating at different sampling rates. Accordingly, the need for efficient conversion between various sampling frequencies has become essential. The process of converting a signal from a given rate to a different one is called *Sampling Rate Conversion*.

First and foremost, it is necessary to remind the reader of the whole process happening before any modification of the signal. Let us consider a music signal created from an instrument (figure 3.1) which is an *analog* signal. If we desire to apply some transformations on this input wave, we could do it in the analog domain but the *digital* one presents more advantages (repeatabilty, noise robustness, etc.). For that purpose, the common procedure would be to use an Analog-to-Digital Converter (A/D). The classical representation of such a converter can be depicted as a Continuous-to-Discrete and Quantizer block. The first one will select a sample every period T and will assign a value to the amplitude of that sample. This acquisition process is a classic A/D conversion. However, they are other ways of performing this acquisition, like in [1].

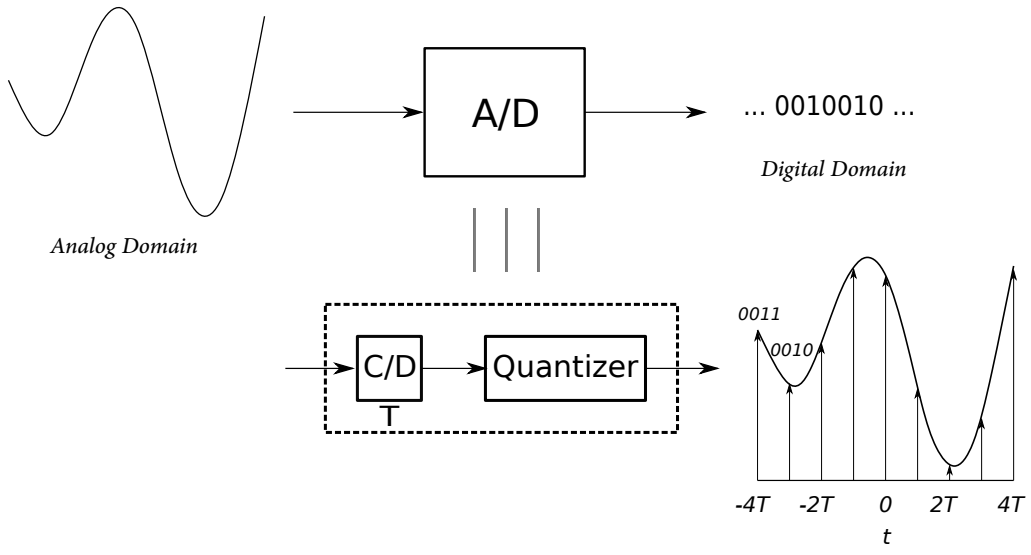


FIGURE 3.1: A/D Converter

Once the signal has been acquired in the digital domain, sampling rate conversion can be applied: we wish to modify the input sample-rate (denoted as $F_{s_{in}}$), namely the one fixed by the C/D converter mentionned above. Indeed, the number of samples picked per second is given by one over T, thus: $F_{s_{in}} = \frac{1}{T}$.

This process can be understood as "resampling after reconstruction" [2] that is reconstructing the sequence of samples as a continuous-time signal and sample it again at a different rate. In theory, two general methods exist to realize the conversion [2], [3]. One method is to pass a given digital signal $x[n]$ through a D/A converter, filter it, and finally pass the analog signal through an A/D converter which will resample the filtered output signal at the desired rate (cf. figure 3.2). The second method is designed to achieve the conversion entirely in the digital/discrete-time domain. In many pratical applications, the latter is prefered. A particularly revelant justification is within the context of audio signal processing in which this project takes place, most of the treatment and interfacing between data streams are carried out digitally. In this way, it makes the interconnections between the different systems easier and more flexible, but also avoids the signal distortion introduced by the D/A and A/D converters.

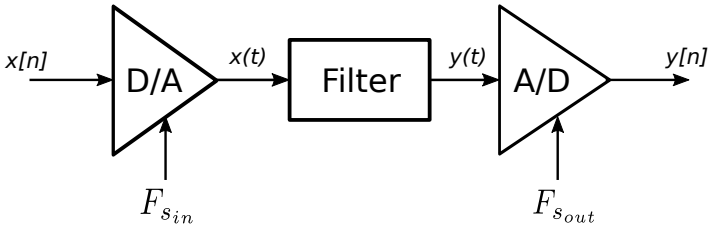


FIGURE 3.2: One Method to Resample a Digital Signal x[n]

Moreover, as presented in the general introduction (see 1), sample-rate conversion can be further sorted into two categories depending on if the relation between the input sampling rate and the output one is known: synchronous and asynchronous. Since the internship of the student was only 6 months, it has been decided that he will only investigate the synchronous part. Nonetheless, the asynchronous method has been largely studied over the pass decades [4]-[6] and clearly encompasses the first type of transformation but could be considered as the continuity of this project.

Hence, this chapter intends to give a valuable description of synchonous sample-rate conversions. Recent studies already attempted to tackle this topic by exploiting the properties of fractional delay filters [7], [8] and filter banks [9], [10]. The approach employed here is more traditional but with the intention to produce an efficient converter in terms of quality and complexity.

3.1.2 Classic Sampling Rate Conversion

The classic method to accomplish a synchronous conversion between two known sample-rates has been amply elaborated in [3]. Hereunder, we will remind the overall principles that rule such a process.

In this particular case, the input and outpout sample-rate $F_{s_{in}}$ and $F_{s_{out}}$ respectively are related by a rational factor $\frac{L}{M}$ i.e. we have: $F_{s_{out}} = \frac{L}{M} F_{s_{in}}$. In order to be able to realize this operation, the following figure presents the method used.

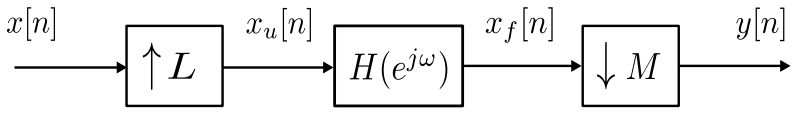


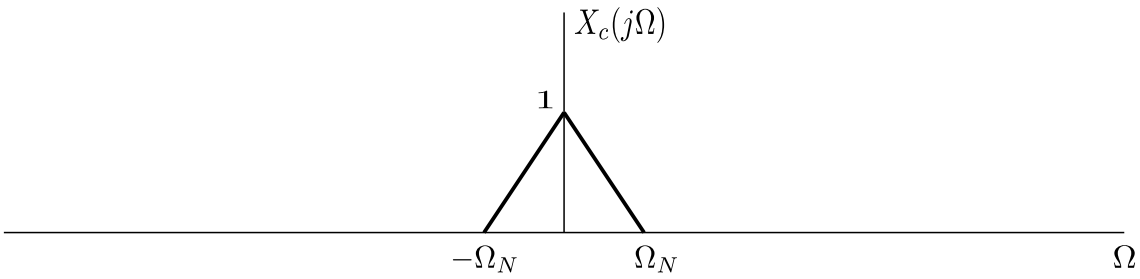
FIGURE 3.3: Block Diagram of Sampling Rate Conversion

The input signal $x[n]$ that we want to resample i.e., change the underlying sampling frequency, is firstly *upsampled*: this block inserts $L-1$ zeros between every sample of $x[n]$. Then the resulting signal $x_u[n]$ is lowpass filtered. Indeed, by interleaving zeros within a signal, it creates replicas of the said signal (see figure 3.4c) that need to be removed otherwise alsiasing will appear in the resampled signal $y[n]$. To finally obtained the desired sampling frequency at the output, the filtered signal is *downsampled* which means that this last block will pick every M^{th} sample of $x_f[n]$ to create the sequence $y[n]$ resampled at the frequency $F_{s_{out}} = \frac{L}{M}$.(frequency of $x[n]$).

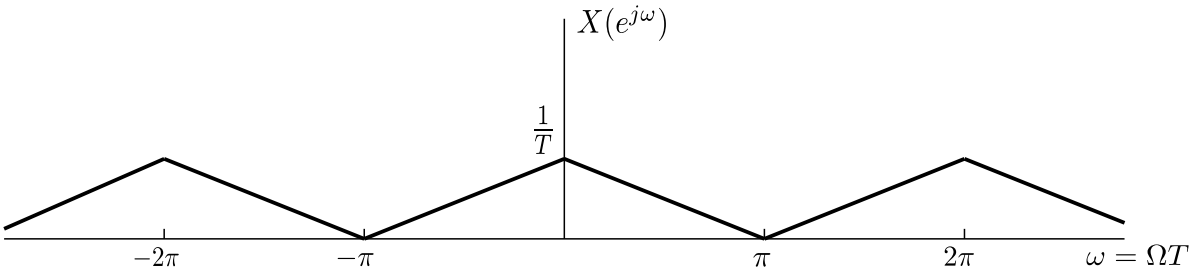
The next figure (figure 3.4) depict the different transitions happening in the continuous and digital frequency domains to a signal when it is resampled by a factor of 2/3. An important consideration to bear in mind is that not every signal can be resampled. According to the *Nyquist-Shannon theorem*, a signal can be fully converted into a digital sequence if the sampling rate is higher than at least two times the maximum frequency contained in that signal. In other words,

$$Fs \geqslant 2 * f_{max} \tag{3.1}$$

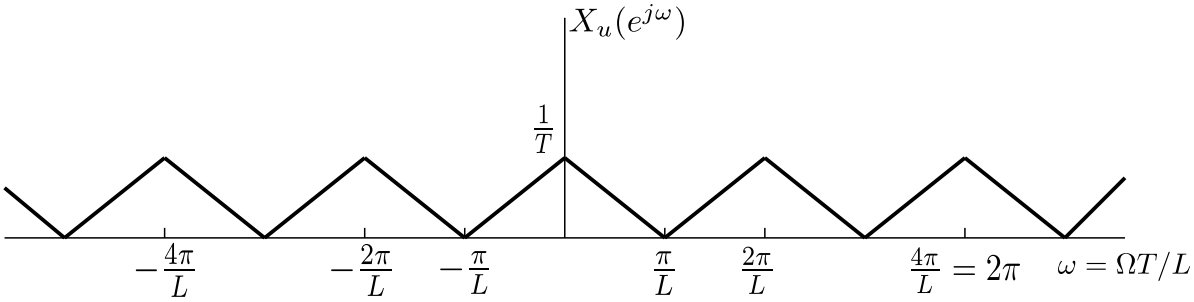
f_{max} is often designated as the *Nyquist frequency* Ω_N.



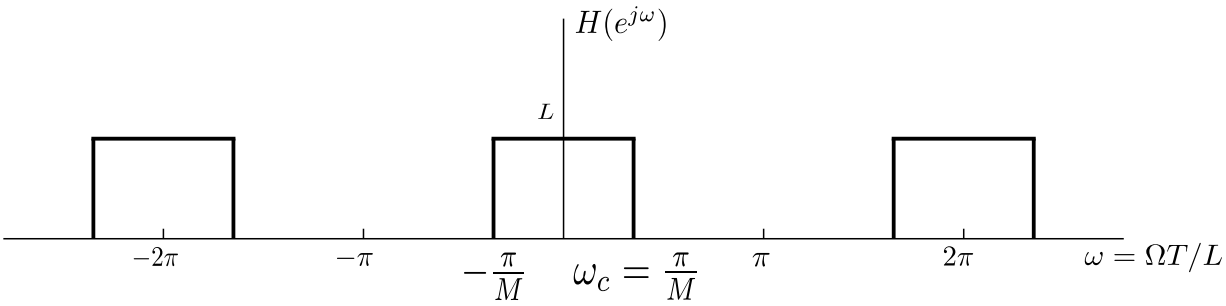
(a) Representation of a Continous Bandlimited Signal



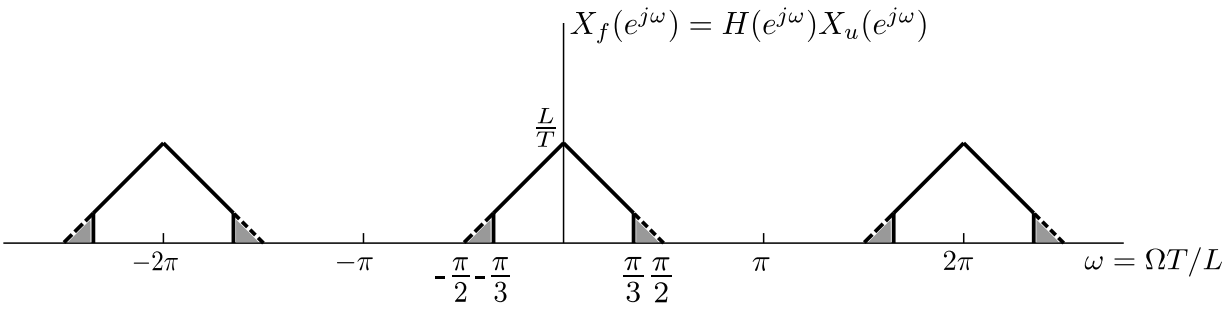
(b) Resulting DTFT



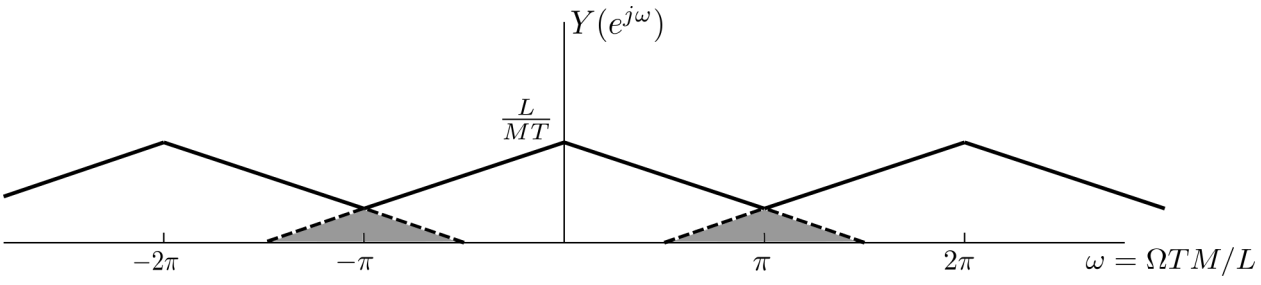
(c) Upsampling by L = 2



(d) Digital Filter used with M = 3



(e) Filtered Signal



(f) Output Resampled Signal

FIGURE 3.4: Sampling Rate Conversion by a Factor of 2/3

A specification can be made regarding the behavior of the conversion depending on the values of L and M:

- if L > M: the sampling rate converter acts as an interpolator i.e. it resamples such as $F_{s_{out}} > F_{s_{in}}$. Thus, no further frequency limitation is needed.
- if M > L: in this case, the converter operates as a decimator i.e $F_{s_{out}} < F_{s_{in}}$. Henceforth, with repect to the Nyquist-Shannon theorem, we must ensure that $f_{max} < F_{s_{out}}/2$. Thereby, the filter $H(e^{j\omega})$ will behave as an anti-aliasing filter and then will remove the extra frequency components in the input signal. This can be noticed in figure 3.4e where the digital frequencies from $\frac{\pi}{3}$ to $\frac{\pi}{2}$ are eliminated. Since $F_{s_{out}} = \frac{2}{3}F_{s_{in}}$, f_{max} is now equal to $\frac{1}{2}.\frac{2F_{s_{in}}}{3} = \frac{F_{s_{in}}}{3}$.

3.1.3 Lowpass Filter

3.1.3.1 Design

The filter previously alluded to is the most important block of the conversion. Because of physical constraints, there is no ideal filter. What follows is an approximation to it and here lies part of the computational complexity challenge.

A filter has several *specifications* or parameters that play a major role in the quality of the obtained filtered signal ($x_f[n]$). First of all, according to the theory [2], [3], the cutoff frequency in the digital domain is given by:

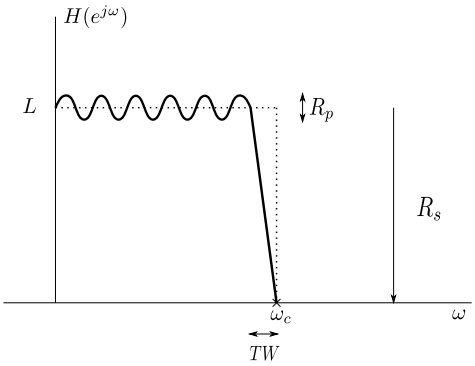
$$\omega_c = \min\left(\frac{\pi}{L}, \frac{\pi}{M}\right) \tag{3.2}$$

The equivalent continous-time frequency is then again:

$$\begin{aligned} f_c &= \left(\frac{LF_{sin}}{2\pi}\right)\min\left(\frac{\pi}{L}, \frac{\pi}{M}\right) \\ &= \left(\frac{LF_{sin}}{2}\right)\min\left(\frac{1}{L}, \frac{1}{M}\right) \end{aligned} \tag{3.3}$$

Furthermore, the gain is also known and equal to L. In fact, if we desire to have a signal at the output with a gain of 1, the equivalent frequency-domain gain must be $\frac{L}{MT} = \frac{1}{T_{out}}$ (cf. figure 3.4f).

Presently, we can introduce the remaning specifications namely, the *ripple in the passband* R_p , *the attenuation in the stopband* R_s and *the transition width* TW. To better grasp these caracteristics, the following figure sums them up.



- R_p and R_s in dB
- TW in percentage of ω_c

FIGURE 3.5: Lowpass Filter and its Specifications

For audio applications, the influence of these parameters is crucial. A high attenuation with a low ripple and a steep transition band will give great quality result at the expense of the complexity of the filter that is, its order will increase. 3.1.2 provides deeper details about this dependency.

From this point forward, what matters is how the filter is implemented. Usually, the common method would be to use a FIR filter because they have a linear phase. It signifies that signals of all frequencies are delayed by the same amount of time, therefore eliminating the possibility of phase distortion which is significant in audio applications. Nevertheless, it seems [11] that a regular listener cannot perceive the effect of phase distortion under special circumstances [12], [13]. Then, it has been decided that the advantages of IIR filters should be also considered. The table below summerizes the main caracteristics of these two families of filters.

Filter	**Order**	**Phase**	**Stability**
FIR	high	linear	always stable
IIR	low	non-linear	can be unstable

TABLE 3.1: Main Differences Between FIR and IIR Filters

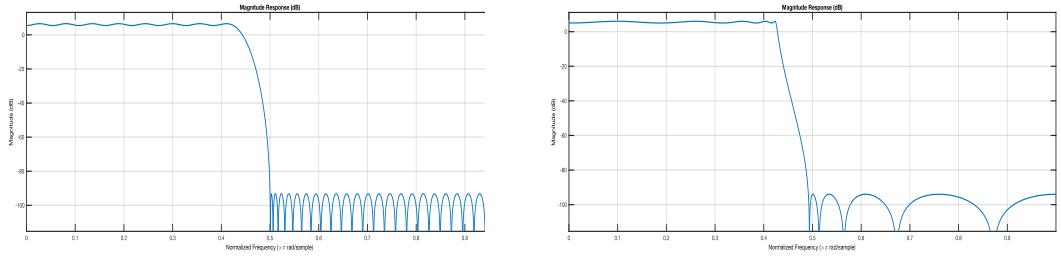
Some analyses have been carried out on what filters have the best performances for the same set of specifications [14]. The Parks-McCLellan implementation was retained for the FIR category and the Elliptic one for the IIR category.

The Parks-McCLellan filters have equiripple in the passband and in the stopband. They minimize the maximum error in both bands for a given weighting function in other words, we have separated control of the deviations in both bands.

The Elliptic or Cauer filters have also equiripple in both bands and usually present the rational function that requires the lowest order for the same specifications. They have four degrees of freedom: order, passband edge, passband and stopband ripple. They are optimal in terms of minimizing the transition bandwidth which is controlled by the order. Below, the magnitude responses of these two filters is provided for (L, M) = (2, 1) and (Rp, Rs, TW) = (1, 100, 0.85).

The filter have been designed in MATLAB. The common procedure [15] requires to specify the normalizing frequency (to transition from continuous to digital frequency). Hereunder, it is the normalizing frequency after the upsampling block L which we call:

$$F_e = LFs_{in} \tag{3.4}$$



(a) Parks-McClellan Filter (b) Elliptic Filter

FIGURE 3.6: Magnitude Responses

3.1.3.2 Efficiency

As well as a conversion of high quality with no aliasing or any other artifacts, an efficient filter, that processes data rapidly is desired. The decision was made that what undoubtedly counts is how many multiplications will be necessary to generate a single sample at the output. In this way, we define the number of *Multiplications Per Output Samples* or MPOS. The lower it is, the faster the conversion will be which is what we need for a real-time process. In the case of a direct implementation of a FIR or IIR filter of order N, we have:

$$MPOS = (N+1)M \tag{3.5}$$

It is conspicuous that if the length of the filter (i.e. the number of its coefficients) is too considerable, the number of MPOS will enlarge resulting in an extremely slow transformation. Thus, the need for improvement would be clearly an obligation.

3.1.4 Improvements

In many cases, the numbers L and M are actually too large, making the computation of the filter's coefficients unfeasible (especially for the FIR case; cf. 3.1). For example, with a conversion from 44.1 kHz to 48 kHz, the resampling factor is then $\frac{L}{M} = \frac{160}{147}$. The cutoff frequency $\omega_c = \frac{\pi}{L}$ will become extremely small, giving less time for the transition to happen and therefore will have the effect of increasing the order. A well known method to address this issue is the polyphase decomposition which permits an easier implementation of the filter. Moreover, the decomposition into several cascaded stages is also explored in this section.

3.1.4.1 Polyphase

The polyphase decomposition is a different way to express the impulse response of the filter. Instead of convolving the signal by the whole filter's IR, only parts of this response are used and they process the input signal simultaneously. It is a parallel representation that can considerably reduce the amount of complexity needed to perform a sampling rate conversion.

The classic method [3], [16] would be to decompose the filter's impulse repsonse $h[n]$ into M (or L) subsequences $h_k[n]$ with $k = 0, 1, \ldots, M-1$ such as

$$h_k[n] = \begin{cases} h[n+k], & n = \text{integer multiple of } M, \\ 0, & \text{otherwise.} \end{cases} \tag{3.6}$$

Now, if we consider the z-transform of the filter,

$$H(z) = \sum_{n=-\infty}^{\infty} h[n] z^{-n} \tag{3.7}$$

we can rewrite it in the following form

$$\begin{aligned} H(z) = & \sum_{n=-\infty}^{\infty} h[nM] z^{-nM} \\ & + z^{-1} \sum_{n=-\infty}^{\infty} h[nM+1] z^{-nM} \\ & \vdots \\ & + z^{-(M-1)} \sum_{n=-\infty}^{\infty} h[nM+M-1] z^{-nM} \end{aligned} \tag{3.8}$$

This can be compactly written as

$$H(z) = \sum_{k=0}^{M-1} E_k(z^M) z^{-k} \tag{3.9}$$

where

$$E_k(z) = \sum_{n=-\infty}^{\infty} e_k[n] z^{-n} \tag{3.10}$$

with

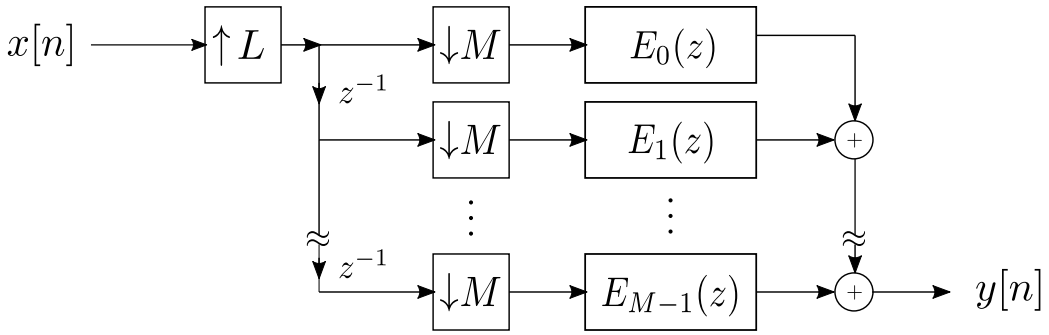
$$e_k[n] = h[nM+k], \quad 0 \leq k \leq M-1 \tag{3.11}$$

The $E_k(z)$ are called the *polyphase components* of $H(z)$, obtained by z-transform [3] of the $e_k[n]$. Equation 3.9 is referenced as type 1 polyphase [17]. An alternative notation can also be presented, named type 2 polyphase:

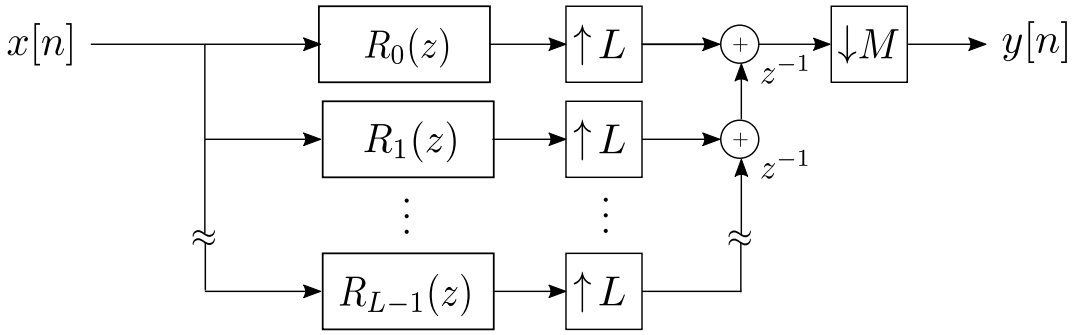
$$H(z) = \sum_{k=0}^{M-1} R_k(z^M) z^{-(M-1-k)} \tag{3.12}$$

where the components $R_k(z)$ are simply a permutation of the $E_k(z)$ i.e. $R_k(z) = E_{M-1-k}(z)$.

According to this notation, we are now able to present a classic representation of a polyphase implementation of a sample-rate conversion system displayed underneath in figure 3.7.



(a) Direct Implementation of Type 1 Polyphase



(b) Direct Implementation of Type 2 Polyphase

FIGURE 3.7: Classic Polyphase Decomposition for SRC Systems

At this point, the reader might wonder why the block M in figure 3.7a or the L one in Figure 3.7b are placed respectively before and after the polyphase components. This has been made possible thanks to the *Noble Identity* introduced in [3, p. 204-205].

These structures considerably reduce the amount of MPOS required for the computation since currently with have $MPOS = \frac{(N+1)M}{M} = (N+1)$ for the type 1 polyphase and $MPOS = \frac{(N+1)M}{L}$ for the type 2 polyphase. However, these arrangements are not the most efficient ones since the polyphase components are not applied at the lowest rate where the hardware complexity is the lowest. This is due to the fact that the expander L and the decimator M are not interchangeable (we cannot apply the noble identity again). Nonetheless, the next section describes what solution has been retained to further enhance our procedure.

3.1.4.1.1 Polyphase Representation for FIR Filters

Before we proceed, it is important to specify that the above polyphase decompositions can only be implemented to a non-recursive filter, that is a filter that only uses the input values to compute its coefficients. Typically, these filters are refered as FIR filters.

For the purpose of having an efficient polyphase structure, as explained in [17], we can exploit the fact that the k^{th} polyphase term in equation 3.9 can be substituted by $z^{-Lka}z^{Mkb}$ with a and b positive integers. As long as L and M are relatively prime (common divisor is 1), the terms a and b can be found according to the Euclid's Theorem since we have $La - Mb = 1$. As of now, every delay z^{-1} in figure 3.7 can be replaced by $z^{-La}z^{Mb}$, thus the expander and decimator can be interchanged which leads to the following stucture displayed on the next page (figure 3.8).

Thanks to this final representation, the MPOS is now decreased:

$$MPOS = \frac{(N+1)M}{LM} = \frac{(N+1)}{L} \tag{3.13}$$

As a result, we obviously chose to use this decomposition. The integers a and b are simply pre-conputed in accordance with the values of L and M by means of the *Euclidean Algorithm*.

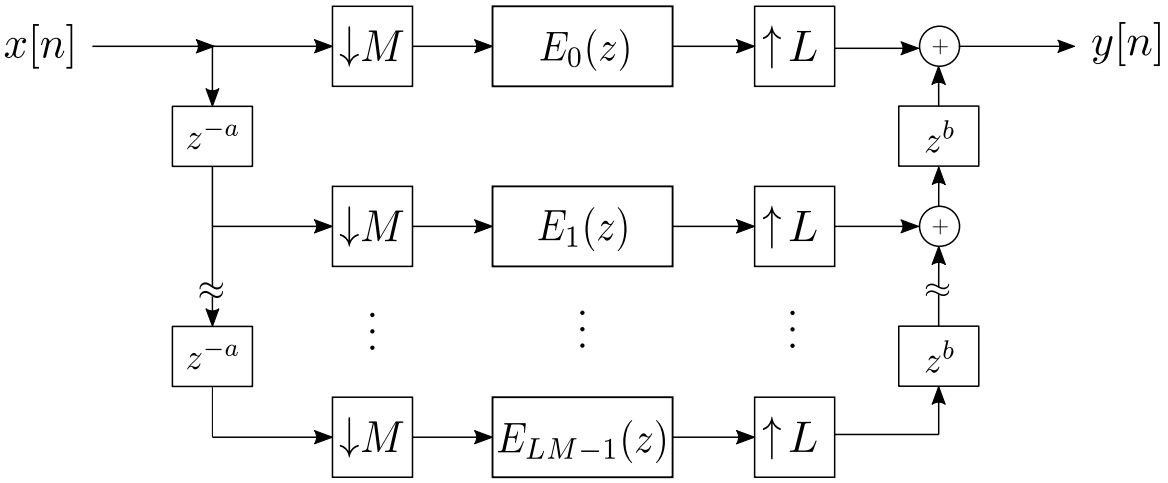


FIGURE 3.8: Efficient Polyphase Decomposition

3.1.4.1.2 Polyphase Representation for IIR Filters

For the IIR filters, the trick is a little bit more complex. As previously described, the former polyphase decomposition can only be applied to a non-recursive filter. Nevertheless, IIR filters are composed of both a recursive and a non-recursive part. Let us take a closer look at its transfer function given by

$$H(z) = \frac{B(z)}{A(z)} = \frac{\sum_{k=0}^{N_z} b_k z^{-k}}{\sum_{k=0}^{N_p} a_k z^{-k}} \tag{3.14}$$

where N_z and N_p are the number of zeros and poles respectively. As we can see, the numerator represents the non-recursive part and the denominator the recursive one. The main goal here is to be able to use the polyphase decomposition presented in figure 3.8. To do so, we need to isolate the non-recursive part. A straightforward approach would be to simply move the numerator $B(z)$ before the expander L such as illustrated in the figure below.

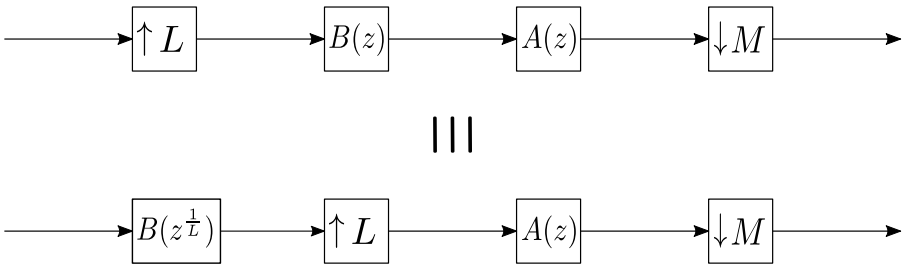


FIGURE 3.9: Direct Polyphase Implementation for IIR Filter

By doing this, the polyphase decomposition can be applied upon the polynomial $B(z^{\frac{1}{L}})$ where the working sampling frequency is lower than at its former position, meaning the complexity is lessened:

$$MPOS = (\frac{N_z + 1}{L} + N_p + 1)M \tag{3.15}$$

Nevertheless, this structure is not the most efficient one since, as before, not all the blocks have been dealt with, that is to say, $A(z)$. An improvement is still possible.

Since $H(z)$ is a fraction of two polynomials, we can factorize them and then rewrite the transfer function in the following way

$$H(z) = k\frac{\prod_{k=1}^{N_z}(1 - z_k z^{-1})}{\prod_{k=1}^{N_p}(1 - p_k z^{-1})} \tag{3.16}$$

with z_k the *zeros*, p_k the *poles* and k the gain. At present, we can have recourse to the substitution

$$1 - p_k z^{-1} \equiv \frac{1 - p_k^D z^{-D}}{\sum_{i=0}^{D-1} p_k^i z^{-i}} \tag{3.17}$$

The next step is to replace equation 3.17 in 3.16 but we need to assign some values to D. The point is to split the poles and raise them to the power of L and M to then be able to use the noble identity. To make it clearer, let's say we have N_z poles and we arbitrarily choose two values N_L and N_M such as $N_L + N_M = N_z$. Now we pick N_L poles γ among the p_k with $D = L$ and the N_M ϵ remaining with $D = M$. The poles we select do not matter, the only important rule to fullfil is to not separate the complex conjugate poles.

Therefore, we can expand equation 3.16

$$\begin{aligned} H(z) &= k\frac{\prod_{k=1}^{N_z}(1 - z_k z^{-1})}{\prod_{k=1}^{N_L}(1 - \gamma_k z^{-1}) \prod_{k=1}^{N_M}(1 - \epsilon_k z^{-1})} \\ &= \frac{k\prod_{k=1}^{N_z}(1 - z_k z^{-1})}{\prod_{k=1}^{N_L}\frac{1 - \gamma_k^L z^{-L}}{\sum_{i=0}^{L-1}\gamma_k^i z^{-i}} \prod_{k=1}^{N_M}\frac{1 - \epsilon_k^M z^{-M}}{\sum_{i=0}^{M-1}\epsilon_k^i z^{-i}}} \\ &= \frac{k\prod_{k=1}^{N_z}(1 - z_k z^{-1}) \prod_{k=1}^{N_L}\left(\sum_{i=0}^{L-1}\gamma_k^i z^{-i}\right) \prod_{k=1}^{N_M}\left(\sum_{i=0}^{M-1}\epsilon_k^i z^{-i}\right)}{\prod_{k=1}^{N_L}(1 - \gamma_k^L z^{-L}) \prod_{k=1}^{N_M}(1 - \epsilon_k^M z^{-M})} \end{aligned} \tag{3.18}$$

To simplify the numerator, we can return to the first definition of $H(z)$ (cf. equation 3.14) and rewrite $k\prod_{k=1}^{N_z}(1 - z_k z^{-1})$ as $\sum_{k=0}^{N_z} b_k z^{-k}$. Then, for the two other products

we have to recall that it is a multiplication of polynomials. In other words, if we consider first the product over N_L, it is a multiplication of polynomials of coefficients $\Gamma_k = [\gamma_k^0, \gamma_k^1, \ldots, \gamma_k^{L-1}]$ for $k = 1, \ldots, N_L$. Thereby, if we name cL the list of coefficients of this product, we end up with $cL = \Gamma_1 * \cdots * \Gamma_{N_L-2} * \Gamma_{N_L-1} * \Gamma_{N_L}$ where $*$ designates the *linear convolution*.

All this leads to

$$\prod_{k=1}^{N_L} \Big(\sum_{i=0}^{L-1} \gamma_k^i z^{-i} \Big) = \sum_{i=0}^{N_L(L-1)} cL_i \, z^{-i} \tag{3.19}$$

where cL_i represents the i^{th} coefficient of the list cL. Likewise

$$\prod_{k=1}^{N_M} \Big(\sum_{i=0}^{M-1} \epsilon_k^i z^{-i} \Big) = \sum_{i=0}^{N_M(M-1)} cM_i \, z^{-i} \tag{3.20}$$

Finally, the transfer function takes the form of

$$H(z) = \frac{\sum_{k=0}^{N_z} b_k z^{-k} \sum_{i=0}^{N_L(L-1)} cL_i \, z^{-i} \sum_{i=0}^{N_M(M-1)} cM_i \, z^{-i}}{\prod_{k=1}^{N_L} (1 - \gamma_k^L z^{-L}) \prod_{k=1}^{N_M} (1 - \epsilon_k^M z^{-M})} \tag{3.21}$$

Anew, this last equation is the product of three polynomials. It simply gives

$$H(z) = \frac{\sum_{k=0}^{N_N} c_k z^{-k}}{\prod_{k=1}^{N_L} (1 - \gamma_k^L z^{-L}) \prod_{k=1}^{N_M} (1 - \epsilon_k^M z^{-M})} = \frac{H_N(z)}{H_L(z^L) H_M(z^M)} \tag{3.22}$$

with $N_N = N_z + N_L(L-1) + N_M(M-1)$ and $c = b * cL * cM$.

The final expression of $H(z)$ is composed of three filters:

1. $H_N(z) = \sum_{k=0}^{N_N} c_k z^{-k}$ which is FIR;
2. $H_L(z^L) = \dfrac{1}{\prod_{k=1}^{N_L} (1 - \gamma_k^L z^{-L})}$, an all-pole filter, function of z^L;
3. $H_M(z^M) = \dfrac{1}{\prod_{k=1}^{N_M} (1 - \epsilon_k^M z^{-M})}$ which is also an all-pole filter, function of z^M.

The structure of the sampling rate conversion is henceforward

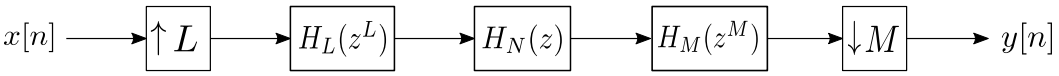


FIGURE 3.10: Decomposition of $H(z)$ as a Cascaded Structure

As depicted in the above figure, the procedure formerly detailed has introduced two filters $H_L(z^L)$ and $H_M(z^M)$ that can be moved respectively accross the expander and the decimator thanks to the noble identity. Furthermore, we can perform the polyphase decomposition seen in figure 3.8 on $H_N(z)$ since it is a FIR filter. It follows

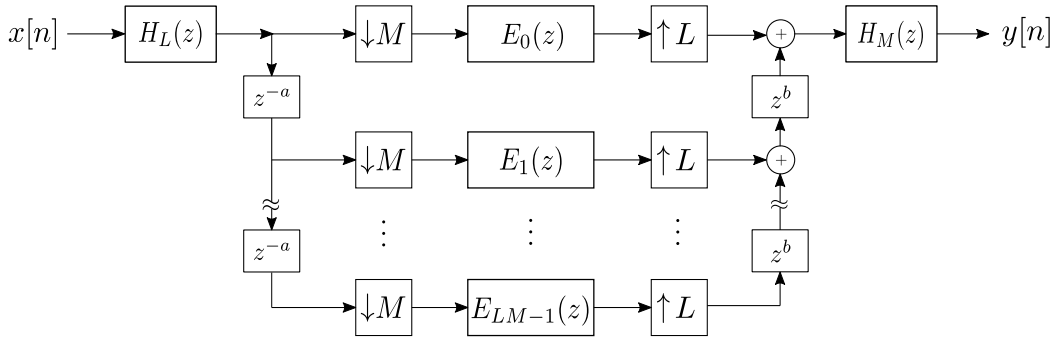


FIGURE 3.11: Efficient Implementation of an IIR Filter for a SRC

This final arrangement is an efficient implementation for IIR filters. It has been introduced in [18]. Although the numerator order is increased by $N_L(L-1)+N_M(M-1)$, the gain in efficiency over the direct implementation is LM for the numerator and $LM/(L+M-1)$ with a number of MPOS of

$$\begin{aligned} MPOS &= N_M + \frac{1}{L}(N_N+1) + \frac{M}{L}N_L \\ &= \frac{1}{L}(N_z+1) + \frac{L+M-1}{L}N_p \end{aligned} \tag{3.23}$$

It appears from this last equation that the number of MPOS does not depend on N_M or N_L, therefore, we can just as well set one of these factors to zero. This method should be prefered over using a FIR filter since the efficiency is greatly enhanced.

3.1.4.2 Multistage

Even so, another improvement is still possible. It is based on dividing the factors L and M into N factors i.e. $L=\prod_{i=1}^{N} L_i$ and $M=\prod_{i=1}^{N} M_i$ where some L_i and M_i can be equal to 1 but are relatively prime $\forall\, i,j$. The goal is to cascade the stages L_i and M_i and compute every filter H_i (cf. figure 3.12). By doing so, the orders are drastically lowered and so are the MPOS.

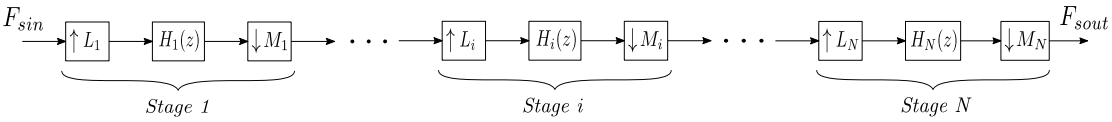


FIGURE 3.12: Multistage Representation

The overall filter can be found again by applying multiple noble identities

$$H(z) = \prod_{i=1}^{N} H_i(z^{\tilde{L}_i \tilde{M}_{i-1}}) \tag{3.24}$$

With the notation $\tilde{L}_i$ and $\tilde{M}_i$ standing for $\prod_{k=i+1}^{N} L_k$ and $\prod_{k=1}^{i} M_k$ respectively and $\tilde{L}_N = \tilde{M}_0 = 1$.

This approach called *multistage* has been widely utilized in interpolators and decimators [2], [3], [17], [19]. It allows less constraints on the filters by reducing their orders and therefore the MPOS is lower than a direct implementation

$$MPOS = (N_1 + 1)M_1 \frac{L_2}{M_2} \dots \frac{L_N}{M_N} + (N_2 + 1)M_2 \frac{L_3}{M_3} \dots \frac{L_N}{M_N} + \dots (N_N + 1)M_N \tag{3.25}$$

with N_i the order of the filter i.

For the case of sample-rate conversion by a fractional factor, two methods on how to design the filters have been investigated in this project:

- The first one may be called the *classic* one. It follows the same theory behind a one-stage system as seen in section 3.1.3. The difference here resides in the calculation of the cutoff frequencies. For each stage i

$$\omega_{c_i} = min\left(\frac{\pi}{L_i}, \frac{\pi}{M_i}\right), \quad i = 1, 2, \dots N. \tag{3.26}$$

  and,

$$H_i(1) = L_i, \quad i = 1, 2, \dots N. \tag{3.27}$$

- The second method has been introduced by *Crochiere* and *Rabiner* in [20] (we will reference it as *second method*). We decided to explore this solution because *Smarc*, an open source project by *Télécom ParisTech* [21] chose this process with FIR Parks-McClellan filters implemented in polyphase decomposition and it gave decent results (cf 3.2.6). To explain the manner to process, we can defferentiate two cases:

  1. $L/M < 1$
     Let's consider the first case when L/M is < 1 that is identified as a decimator. The way to proceed is to separate the system into several blocks D_K depicted in the figure 3.13 with

$$D_i = \frac{M_i}{L_i} > 1, \quad i = 1, 2, \dots K. \tag{3.28}$$

     and,

$$f_{ri} = \frac{f_{r(i-1)}}{D_i} = \frac{L_i}{M_i} f_{r(i-1)}, \quad i = 1, 2, \dots K. \tag{3.29}$$

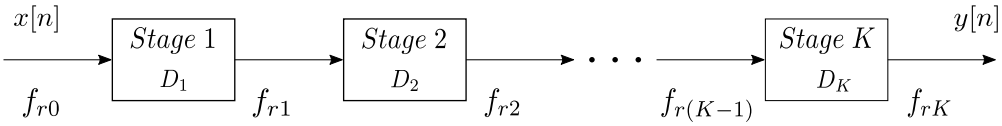


FIGURE 3.13: Representation of K-stage Decimator

The initial sampling rate is f_{r0} and the final one is f_{rK} with $f_{r0} > f_{rK}$. The global reduction factor is $D = \frac{f_{r0}}{f_{rK}} = \prod_{i=1}^{K} D_i$, greater than 1. A block D_i is equivalent to a stage i presented in figure 3.12, the difference lies in the design of the filters: the stopband of the filters has to be adapted for each stage without modifying the passband. The way to do so is sketched as follows

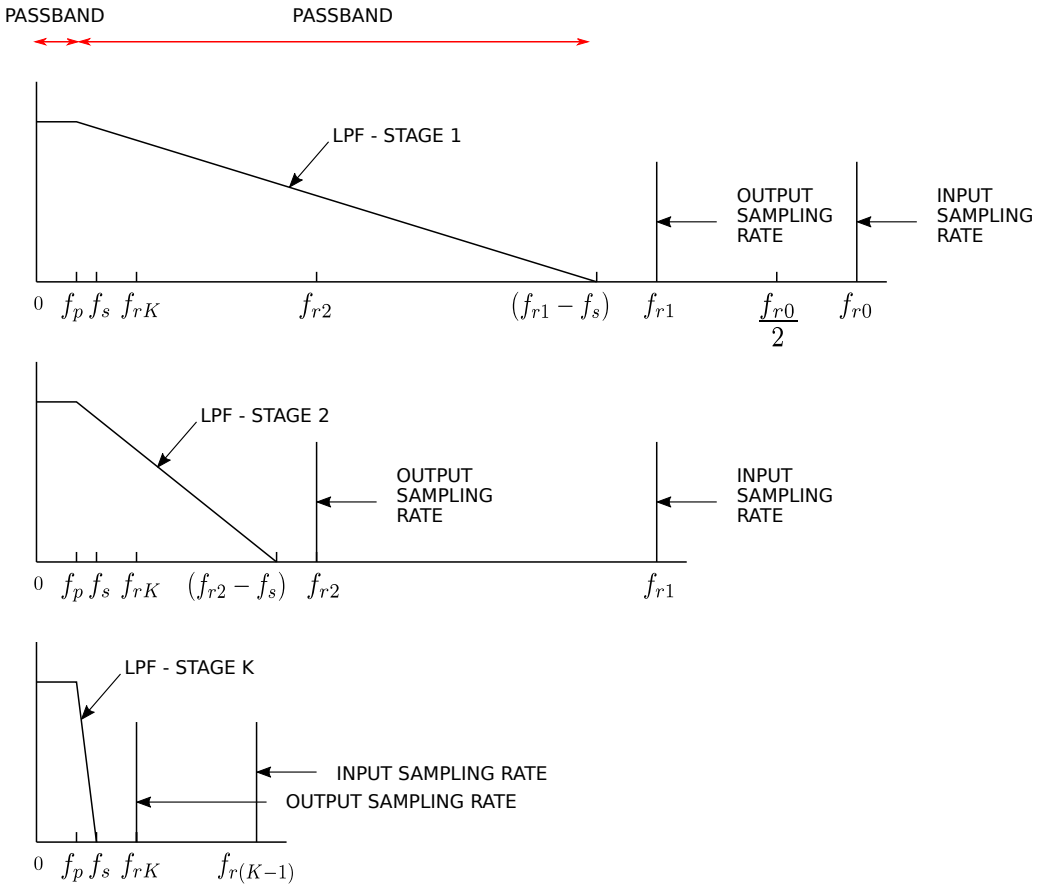


FIGURE 3.14: Frequency Response Interpretation of the K-stage Decimation Process

f_s is the maximal frequency in the signal through each stage

$$fs \leq \frac{F_{rK}}{2} \tag{3.30}$$

It can be noticed that the passband never changes but the stopband is. However the equivalent digital passband frequency $\omega_{p_i} = f_p / f_{ri}$ is different for every stage. A thorough method to find the most optimized D_i has been demonstrated in [20] for a chosen number of stage K. Nevertheless, it

depends on abacuses and always result in approximating their values since they need to be integers. What has been shown in [19] is the importance of the number of stages on the overall efficiency: for $K = 3$ or 4, the MPOS is the lowest. Having this in mind, the values of L_i and M_i for a given ratio L/M with this amount of stages are often unique. Thus, it may not be worthwhile to use this tedious method of choosing the D_i, but simply pick them manually, especially if we choose a high number of stages.

2. $L/M > 1$
   For the case of $L/M > 1$ that is, the case of interpolation, the reasoning is the same. This time

$$f_{r(i-1)} = D_i f_{ri}, \quad i = 1, 2, \ldots K. \tag{3.31}$$

with $D_i = \frac{M_i}{L_i} > 1$ (the role of M_i and L_i are reversed). Since interpolation is the dual of decimation, we have

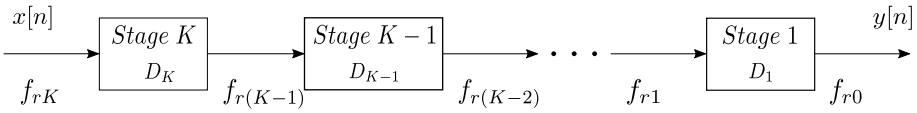


FIGURE 3.15: Representation of K-stage Interpolator

and the filter design in the frequency domain

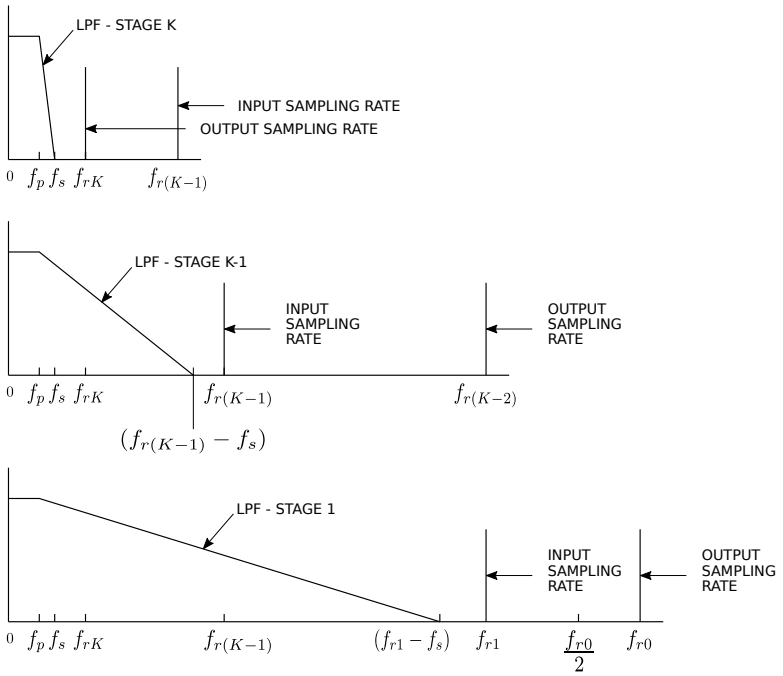


FIGURE 3.16: Frequency Response Interpretation of the K-stage Interpolation Process

The same comment made in regard to how to find the D_i applies here.
Although these two methods are different design-wise, they share a common feature. The passband ripple has to be altered for both cases. If we desire the total passband (after combination of all the stage filters) to be

the same as chosen originally by the filter specifications, we must now have $R_p = \frac{R_p}{\text{Number of Stages}}$. However, the stopband attenuation remains unchanged (see [2]). Moreover, equation 3.27 is still valid for this approach. In conclusion, what remains is comparing these two methods and discover which one gives best quality and efficiency result (MPOS).

One could wonder what the influence of the organization of the stages is, in other words, which L_i and M_i are best for which stage. But let us present an example to buttress our concern.

In this example, we consider the case where $L = 9$, $M = 8$ and a number of stages $N = 2$. Since, $L_1 = L_2 = 3$, the result is two possible implementations of a dual stage SRC.

- **Case** $M_1 = 2$ **and** $M_2 = 4$

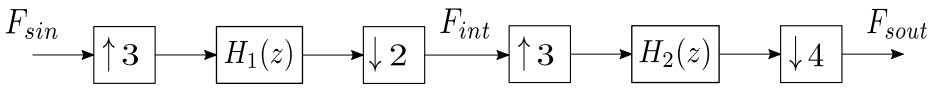


FIGURE 3.17: Corresponding 2-stage decomposition

The cutoff frequencies of H_1 and H_2 are $\pi/3$ and $\pi/4$, respectively. We have $H(z) = H_1(z^3)H_2(z^2)$. As the following plot shows, there should be no problem :

- outside of low frequencies, the filters act on different frequency bands,
- the resulting cutoff pulsation is $\pi/9$ as expected.

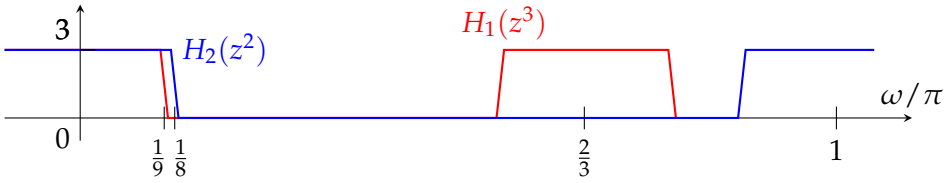


FIGURE 3.18: Frequency Representation of Resulting Filters

- **Case** $M_1 = 4$ **and** $M_2 = 2$

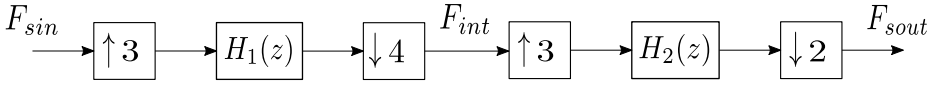


FIGURE 3.19: Corresponding 2-stage decomposition

The cutoff frequencies of H_1 and H_2 are $\pi/4$ and $\pi/3$, respectively. We have $H(z) = H_1(z^3)H_2(z^4)$. As the following plot shows, the situation is now more complicated because

- the filters share common frequency intervals in high frequencies,
- the resulting cutoff pulsation of $\pi/12$ is lower than expected.

What can be retained from this example is that if the first stage is composed of the factors L_1 and M_1 such that $M_1 > L_1$, this will result in a degradation of the signal. More generally, we can extract two rules out of these examples:

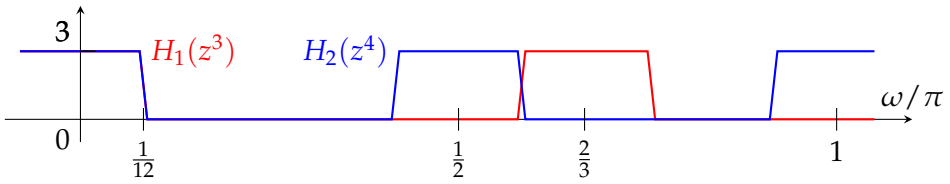


FIGURE 3.20: Frequency Representation of Resulting Filters

- if $L > M$ ($F_{s_{out}} > F_{s_{in}}$), to avoid alisasing, every intermediary frequencies F_{int} must not be smaller than $F_{s_{in}}$.
- if $M > L$ ($F_{s_{in}} > F_{s_{out}}$), F_{int} must not be smaller than F_{sout} to avoid alisasing.

The section 3.2.2 will delve into how properly organize the stages while considering the above aspect.

3.1.4.3 Multistage and Polyphase

Just as polyphase, multistage decomposition can improve how to decrease the number of Multiplications Per Output Sample which is crucial if we want to process data in real-time. Evidently, we can combine these advancements.

- For FIR filters, the final structure would be a combination of figures 3.12 and 3.8 for evey stage i. Moreover we can use the property that these filters can be implemented as a symetric structure called *folded structure* which will divide the direct MPOS (cf. equation 3.5) by 2. Thus, the final number of multiplications is

$$MPOS = \frac{(N_1+1)M_1}{2L_1}\frac{L_2}{M_2}\cdots\frac{L_N}{M_N} + \frac{(N_2+1)M_2}{2L_2}\frac{L_3}{M_3}\cdots\frac{L_N}{M_N} + \cdots + \frac{(N_N+1)M_N}{2L_N} \tag{3.32}$$

- For IIR filters, the final structure would be a combination of figures 3.12 and 3.11 for evey stage i. Therefore, we have

$$\begin{aligned} MPOS = &\left(\frac{1}{L_1}(N_{z_1}+1) + \frac{L_1+M_1-1}{L_1}N_{p_1}\right)\frac{L_2}{M_2}\cdots\frac{L_N}{M_N} + \\ &\left(\frac{1}{L_2}(N_{z_2}+1) + \frac{L_2+M_2-1}{L_2}N_{p_2}\right)\frac{L_3}{M_3}\cdots\frac{L_N}{M_N} + \\ &\cdots + \left(\frac{1}{L_N}(N_{z_N}+1) + \frac{L_N+M_N-1}{L_N}N_{p_N}\right) \end{aligned} \tag{3.33}$$

  In this case, if we apply the same folded structure principle to the resulting FIR part $H_N(z)$ in equation 3.22, the number of MPOS would depend on N_M and N_L since

$$
\begin{aligned}
MPOS = &\left(\frac{(N_{z_1}+1)}{2L_1} + \frac{N_{M_1}(2L_1+M_1-1)+N_{L_1}(2M_1+L_1-1)}{2L_1} \right) \frac{L_2}{M_2} \cdots \frac{L_N}{M_N} + \\
&\left(\frac{(N_{z_2}+1)}{2L_2} + \frac{N_{M_2}(2L_2+M_2-1)+N_{L_2}(2M_2+L_2-1)}{2L_2} \right) \frac{L_3}{M_3} \cdots \frac{L_N}{M_N} + \\
&\vdots \\
&+ \left(\frac{(N_{z_N}+1)}{2L_N} + \frac{N_{M_N}(2L_N+M_N-1)+N_{L_N}(2M_N+L_N-1)}{2L_N} \right)
\end{aligned} \tag{3.34}
$$

Minimizing this last equation might be more difficult than minimizing equation 3.33. Thus, the latter should be the one to consider.

3.1.5 Conclusion

The common methods to realize a sample-rate conversion by a rational factor L/M have been studied through this part. The classical principle would be to select an expander L, a low-pass filter and a decimator M. Multiple filters could be used and usually FIR ones because of their linear phase. Nevertheless, the choice has been made to utilize a FIR Parks-McClellan filter and an IIR Elliptic one to compare the quality of the conversion and their efficiency in terms of MPOS. Some improvements have to be considered such as polyphase and multistage decompositions if real-time performance is required. The next part will now focus on a real case: implementing and comparing the previous statements in order to confirm or disconfirm them.

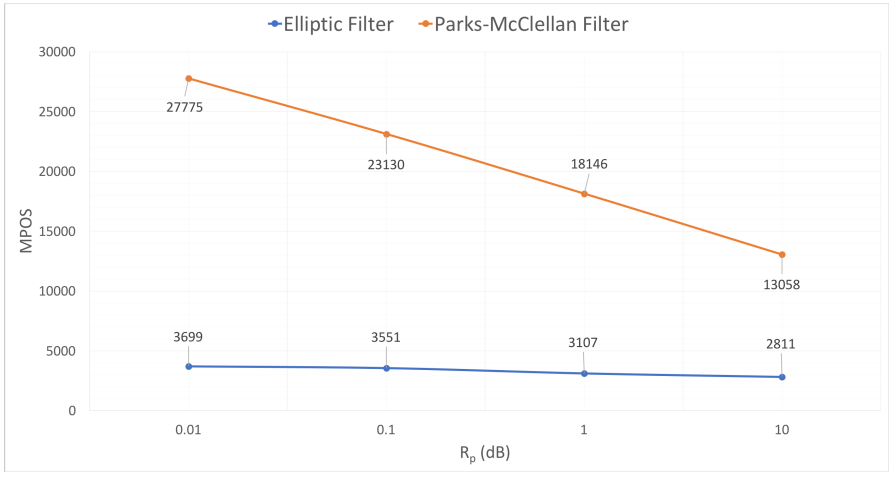
3.2 Study Case

In this section, we will apply the theory described earlier to a tangible example. We decided to study the case of converting from 44.1 kHz to 48 kHz and vice versa. The methods explained in this part are applicable to any rational sampling rate conversion ratio. However, for illustration and practical purposes, we will focus on this particular case.

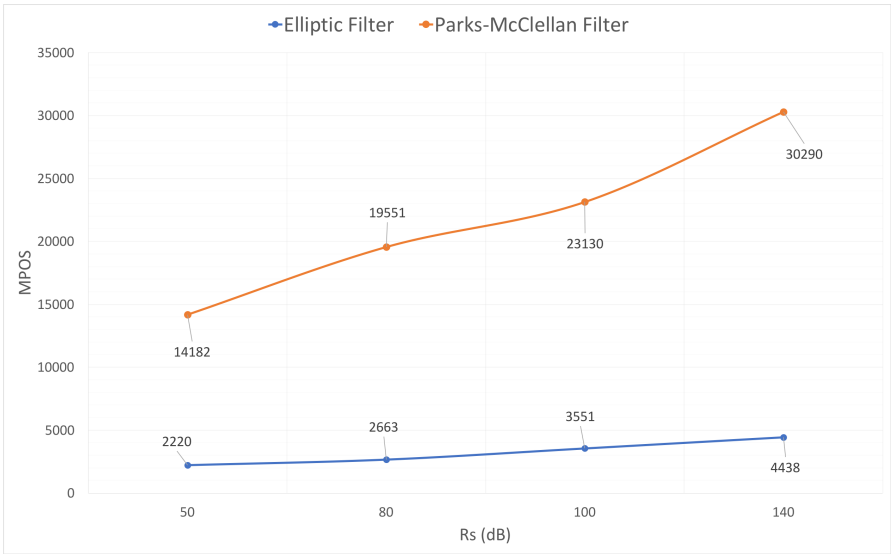
3.2.1 Filter Specifications

As already stated, this conversion requires a ratio of 160 over 147 and if we use filters with low error tolerances (with regard to an ideal lowpass filter), the order is in general high thus imposing increasing the computational complexity and making it more challenging to develop implementations suited for real-time operation. Therefore, reducing this complexity is paramount. In this way, some simplifications have to be made. However, it is desirable to have a good approximation of an ideal low-pass filter because it determines the quality of the conversion.

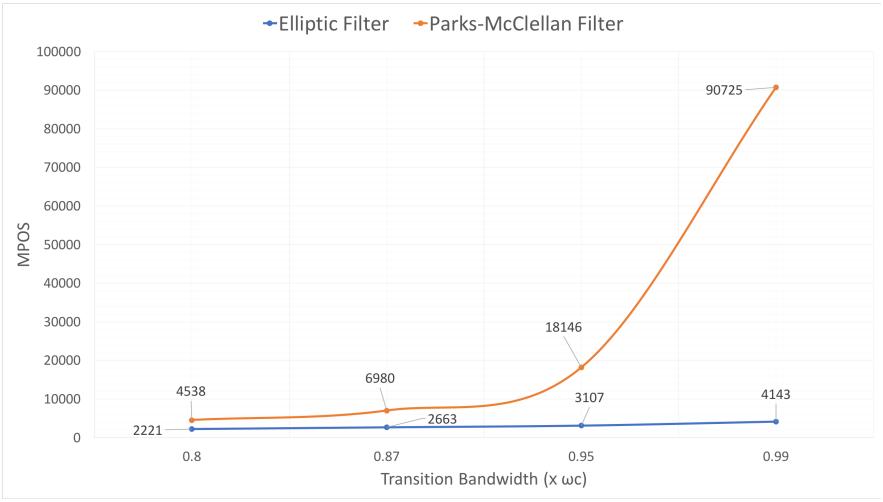
Hence, a first concern was the effect of the different parameters R_p, R_s and TW on the order of the filters, or in other terms, on the MPOS. The following graphs have been made for both Parks-McClellan and Elliptic filters implemented with the polyphase configurations 3.7b (type 2) and 3.9 respectively. These implementations have been chosen for the sake of simplification. What really matters here is the effect of the parameters on the MPOS, not their values strickly speaking.



(a) Impact of the Ripple in the Passband R_p on the *MPOS*



(b) Impact of the Ripple in the Stopband R_s on the *MPOS*



(c) Impact of the Transition Width *TW* on the *MPOS*

FIGURE 3.21: Impact of the Filters' Parameters on the Number of MPOS

First and foremost, as already stated in section 3.1.3, the number of MPOS is definitely more important for FIR Parks-McClellan than IIR Elliptic. However, we observe quite the same behavior: in figure 3.21a, the amount of multiplications is divided by 2 if R_p is within the range $[0.01 - 10]$ dB for the FIR case and slightly less for the IIR one. We can observe the opposite behavior when considering the attenuation R_s within the range $[50 - 140]$ dB. Finally, in figure 3.21c, important differences between the two filters can be noticed about the effect of the transition width. It multiplies the MPOS by approximatively 2 for the Elliptic case and by nearly 20 for the Parks-McClellan filter.

Here again, we have another example that is Elliptic filters should be prefered over Parks-McClellan because of the smaller number of MPOS in general. One may want a good approximation of an ideal low-pass filter and then choose strong parameters therefore resulting in a slow process. In this case, we should opt for increasing the transition bandwidth, giving more freedom to change the other paramters without impacting too much the MPOS, at least for the FIR case.

3.2.2 Multistage Combinations

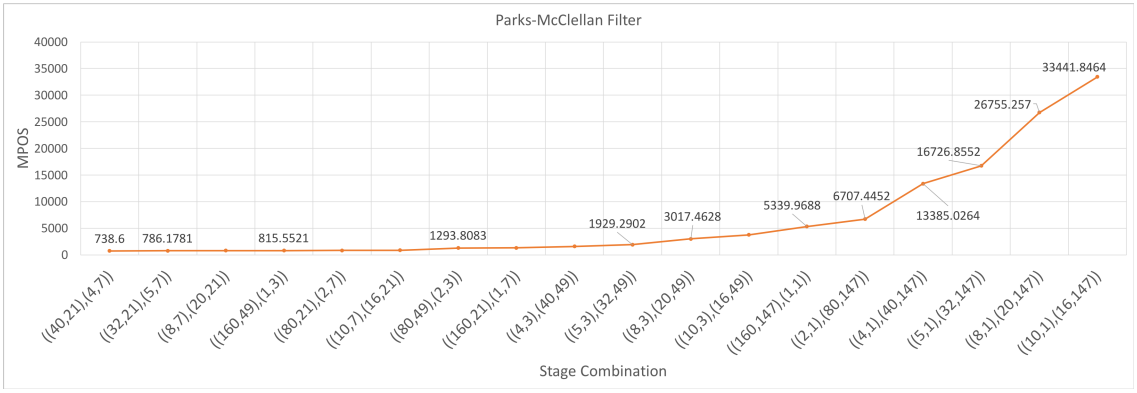
A major investigation made in this part was the effect of multistage and how to develop it in the most efficient way. In [2], [17] and [3], it is claimed that multistage allows reduction of the overall number of MPOS by separating L and M into several smaller factors L_i and M_i and thereby putting less constraints on the design of the ones corresponding to the multistage implementation. However, as demonstrated in [19], the order of the stage factors plays an important role as well as the number of stages.

The goal of this study case is to minimize the equations 3.32 and 3.33. If we consider the first equation, one could think about minimizing the first factor of the sum $\frac{(N_1+1)M_1}{2L_1}\frac{L_2}{M_2}\cdots\frac{L_N}{M_N}$ because the cumulative product of the L_i/M_i is most important in this factor. We already know from the examples in 3.1.4.2 that some combinations are not possible and especially for a conversion from 44.1 kHz to 48 kHz, M_1 cannot be higher then L_1. Therefore, we could choose the smallest M_1 factor possible and the biggest L_1. However, since the stage factors are related ($L = \prod_{i=1}^{N} L_i$; $M = \prod_{i=1}^{M} M_i$), minimizing one summation factor in 3.32 might increase another, resulting in an suboptimal number of MPOS.

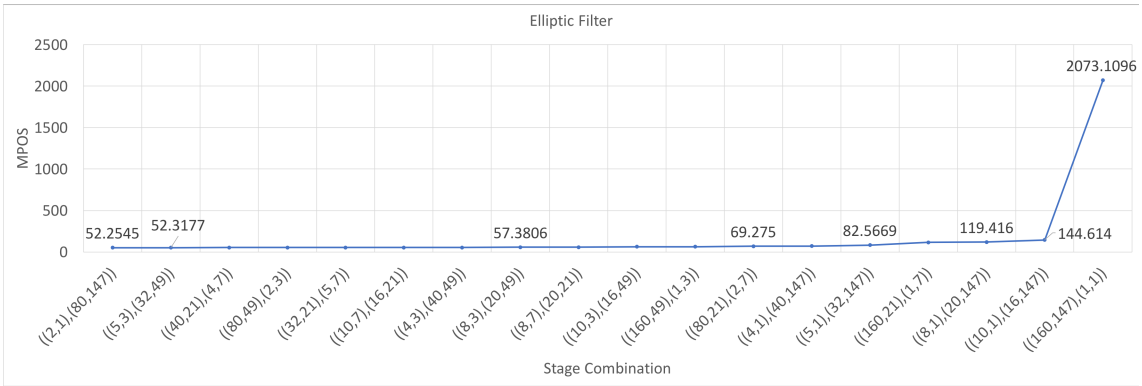
For the second equation (3.33), the weight of the L_i and M_i is harder to determine. However, in the case of the Elliptic filter, we have $N_{z_1} = N_{p_1}$, we could then think of choosing $L_1 \gg M_1$. As stated previously, this decison might impact the other factors and end with poor efficiency.

In the interest of clarfying the above observations, the graphs herein (cf. figure **??**) present the different combinations possible for 2 stages. Each combination of $((L_1, M_1), (L_2, M_2))$ has been optimized with MATLAB, that is, there is no better arrangement (or permutation) for these specific values of L_i andM_i. [1]

[1] Combination $((160, 147), (1, 1))$ is irrelevant because it adds a extra stage to the direct conversion i.e. $MPOS((160, 147), (1, 1)) > MPOS((160, 147))$. It is present in the graphs just to ensure that every combination has been taken into account.



(a) Implication of Stage Combinations on Parks-McClellan Filters



(b) Implication of Stage Combinations on Elliptic Filters

FIGURE 3.22: Implication of Stage Combinations on the Number of MPOS

First of all, the two curves may have an exponential growth, in other words, some combinations make the number of MPOS grow drastically and suddenly. Morevover, as predicted in the theoritical section, the number of MPOS of the Elliptic filter is lower than the one of the Parks-McClellan filter. There is only one combination $(((10,7),(16,21)))$ that is located at the same postiion on both graphs. The contrast is again due to the difference of the MPOS equations.

- Parks-McClellan Filter: the combination that minimizes the number of MPOS is, as suspected, not the one with the smallest M_1 and highest L_1, but the combination $((40,21),(4,7))$, which gives a number of MPOS of 738.6. The combination with such factors is $((160,21),(1,7))$ with $MPOS = 1326.4688$. It seems rather intricate to extract a method on how to minimize the cost of computation by simply exploiting this first graph. But if we take a look at some combinations made for four stages in Appendix A, we should be able to extract a pattern. What appears to reduce the cost of computation is when the factors L and M are well distributed over the different stages, thus giving stage combinations with small L_i and M_i. For instance, the difference between the first and second lines in table A.1 is factor 8 who is not split in $8 = 4.4$ in the second combination.

  To sum up, what should be advocated while designing a multistage implementation of Parks-McClellan filters is that the element L and M should be

divided into close numbers. If it is not possible (as in a two-stage implementation), what is important is to pair the L_i and M_i which are in the same order of magnitude. Then, among the pairs we have made, select the one with a relatively high L_1 and low M_1 while still respecting $M_1 < L_1$. [2]

- Elliptic filter: the same reasoning can be applied here. The pairs have to be similar numbers. However, it is more complex to find a pattern in this case. Indeed, the pair that minimize the equation 3.33 is $((2,1),(80,147))$. It seems that this combination tries to completely avoid lending weight to the second factor of the equation. The next sections will confirm or refute this theory.

3.2.3 Number of Stages

Hence, we are able to order the stage factors in such a way that decreases the cost of computation as much as possible. However, there are certainly a number of stages that are more advantageous than the others.

The following table illustrates the effect of the number of stages on the efficiency. The combinations used are those with respect to what has been previously detailed.[3] The filter's specifications are $(R_p, R_s, TW) = (0.1, 140, 0.85)$.

Filter Structure / Number of Stages	Parks-McClellan Folded	Elliptic Russell
1	5048.9906	30.7063
2	738.6	52.2545
3	310.4333	60.1405
4	255.9408	64.5922
5	261.3568	*N/A*
6	265.9228	*N/A*

TABLE 3.2: Difference of MPOS in the Number of Stages

It is clear from the above table that spreading the implementation over multiple stages considerably reduces the complexity of the design. In the case of the use of a Parks-McClellan filter, four stages is the best option; dividing the number of MPOS by almost 20 if we consider one stage as a reference.

Surprisingly, this result is not applicable for the case of an Elliptic filter implemented as described in [18]. A single stage implementation gives the lowest complexity and for a number of stages larger than four, the design is even not possible. This is due to a too large large attenuation (R_s) combined with a too small ripple (R_p) (divided by 5 or 6 in these cases). We could reduce the attenuation in the stopband and it would work this time. However, a single stage will always have the lowest computational cost, whatever the filter's specifications may be.

Nonetheless, this outcome serves our interests because a single-stage implementation is definitely simpler than a multistage one. Moreover, it explains why the combination $((2,1),(80,147))$ was the best one for two stages. Indeed, the "heavy" second pair is trying to minimize the effect of the second factor of the equation.

[2] It explains why the combination ((8,7),(20,21)) which has really similar factors in each stage is not the most efficient combination. This arrangement forces us to select a suboptimal L_1.

[3] The use of polyphase decomposition was also used.

3.2.4 Use of Minimum-Phase Filters

This study provides guidelines on how many stages to choose and on how to pair the factors. It is useful to mention that an extra comparison has been made. Indeed, in [19], the author claims that he obtained the lowest cost by combining an elliptic filter as a first stage and three *Schuessler* filters in three consecutive stages. We tried to recreate this result.

The first stage is an elliptic filter designed in the same way as in figure 3.11. The rest of the three stages (name Schuessler filters) are minimum-phase filters, that is Parks-McClellan filters where all the zeros are within the unit circle. This method allows to transform FIR low-pass filters with linear-phase into minimum-phase FIR filters of half the degree, while maintaining the same passband and stopband edge frequency characteristics. Halving the degree will reduce the complexity. The technique utilized is detailed in [19] and is possible because the zeros of linear-phase FIR filters appear on the unit circle or in symmetric pairs about the unit circle. The overall MPOS of this approach are **129.2269**. Thus, this new combination allows to have a cost lower than simply combining Parks-McClellan filters. Nevertheless, it is not the most efficient one since the one-stage elliptic filter implemented as A. Russell described it ([18]) is more efficient. This divergence might be explained by the fact that in the case of [19], the author implemented a multistage interpolator (use of expander factors L_i only) and not a full synchronous sample-rate converter. Moreover, he also did not use the method suggested in [18].

3.2.5 Assessment of the Quality of the Conversion

Heretofore, the emphasis was put on the efficiency of the conversion and more particularly, on how to reduce as much as possible the number of MPOS. What remains from now on is to assess what we defined as the quality of the conversion, i.e. detecting any artifacts at the output. In order to do so, we decided to use a common method of sweep frequency response analysis. A frequency sweep is sent through the system and the output is then compare to the original input. A similar response will provide evidence that no aliasing occured. However, this kind of measurement cannot be sufficient. Indeed, this method does not provide any information about the impulse response of the overall system as well as its phase response. There could be some ringing, which is undesirable in audio applications.

The systems evaluated here are a four-stage Parks-McClellan filter, a single stage *Russell* Elliptic filter and the combination of the former filter with three Schuessler filters.

The input is before any expander L or L_i and the output is after the last processing block (decimator M or M_i or filter in the case of Russell). The sweep sent is within the frequency range of 20 Hz - 20 kHz, which is the frequency range of human hearing.

For instance, the following outputs (figure 3.24) are those of the two filter structures used earlier with the specifications $(R_p, R_s, TW) = (0.1, 140, 0.85)$ and implemented in the most efficient form (red underlined MPOS in table 3.2).

As we can observe, the two graphs are identical to the input (cf. figure 3.23) which implies that the conversions processed the signal as expected without generating any aliasing or other artifacts. This procedure cannot be used to make a go/no-go decision about the usability of a device. It can only provide a "no-go" - or indications of simple errors in an audio transmission system. The windowing

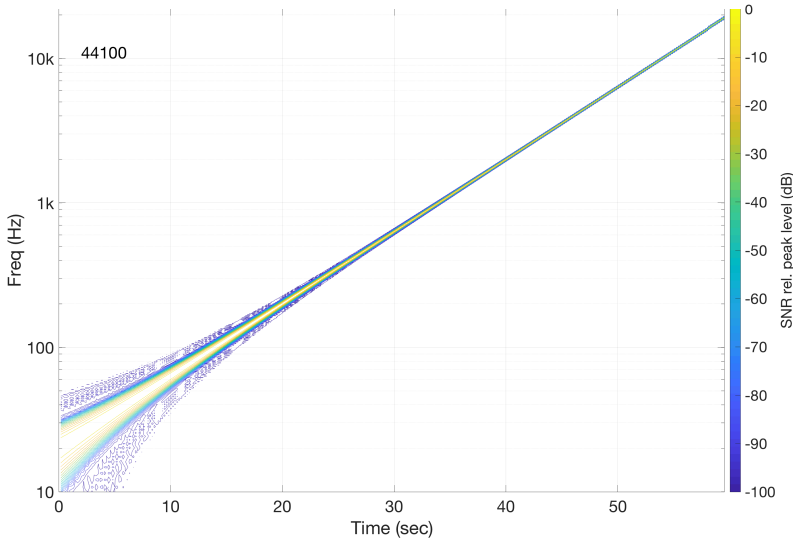
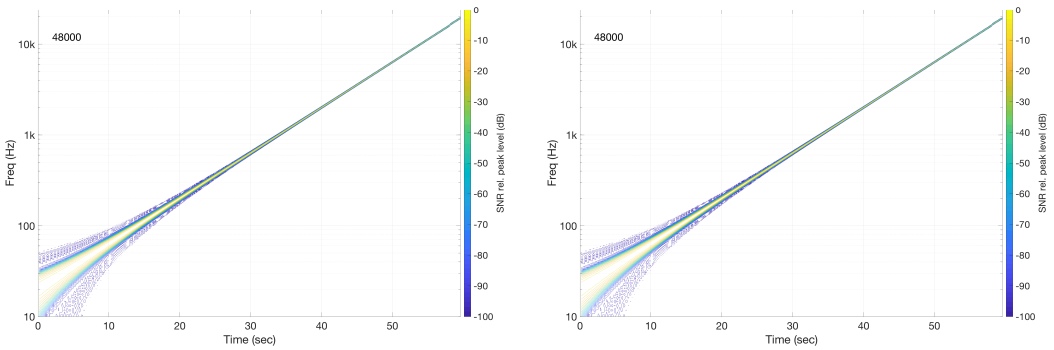


FIGURE 3.23: Frequency Sweep Input



(a) Folded Parks-McClellan Filter (b) Russell Elliptic Filter

FIGURE 3.24: Sweep Frequency Response Analyses of the Two Filter Structures

function in the analysis means that we do not have precision at low frequencies. Consequently, it results in the spreading in the low frequencies. This is an artefact of the analysis, not the system under test. A colored scale is displayed on the right which gives the level of attenuation of the SNR in dB. Thus, without considering the artifacts of the analysis, the yellow straight line indicates that there is no attenuation of the signal while going through the system.

These analyses have been made with certain parameters:

- Bit depth: it is the resolution of the sample, that is how many bits are used for each sample. In our case, 16 bits have been chosen. Testing with 24 bits does not bring any differences to the output graph.
- dBFS: It is used for amplitude levels in digital systems. The level of 0 dB FS is assigned to the maximum possible level. A signal that reaches 50 percent of the maximum level would, for example, have a value of -6 dB FS. In our case, there is no differences at the output between a signal of 0 dBFS and -1 dBFS.
- Bitrate: it is the number of bits transmited per second. We tested for 128, 256 and 320 kbits per second and did not noctice any difference.

If we apply the same procedure to the method involving the three Schuessler filters, what is shown in figure 3.25 is the output of this precise system.

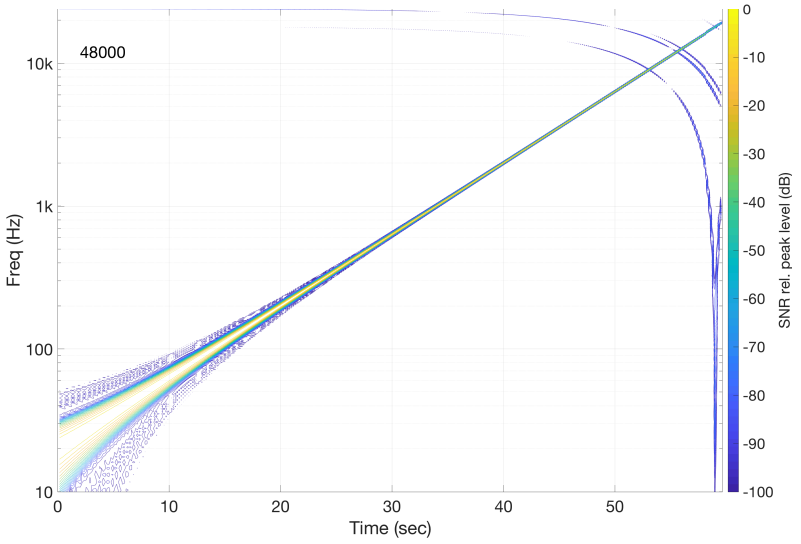


FIGURE 3.25: Sweep Frequency Response Analysys of the Schuessler Filters Combination

This time, aliasing is noticeable at high frequencies. These blue lines are some copies created after the signal was upsampled (cf. figure 3.4c). The combination of the Schuessler filters does not completely filter them out. Therefore, the conversion is flawed.

Even by refining the filter specifications (reducing the transition width for instance), the outcome is still the same. Consequently, we recommend to avoid using this technique and instead stick to the four-stage Parks-McClellan case despite the fact it is less efficient, as at least, it has a better quality of conversion and that is what we are mainly concerned with.

3.2.6 Comparison Multistage Design

One major investigation completed was the way to design the filters in a multistage configuration. As explained in the section 3.1.4.2, several approaches exist and two were considered. The first one was the classic way detailed in 3.1.2 and the second, described in [21] and utilized in the open-source project called *Smarc*. The project took consideration of this method because it implements Parks-McClellan folded filters in a polyphase and four-stage configuration, which is one of the method retained by this project. Moreover, it is one of the few available plugins on internet that use this method. The others are using classic windowed sinc filter methods which are not optimal. The next graph (figure 3.26) is the quality evaluation of the plugin.

It has been realized with the following specifications: $(R_p, R_s, TW) = (0.1, 140, 0.95)$. We can notice some artifacts: Other lines parallel to the fundamental indicate harmonics generated by periodic distortion and additional noise (vertical stains at the end of the sequence). Indeed, there are numerous errors but their effect is low in terms of decibel if we take a look at the scale (between -90 and -100 dB). Therefore, it is still worth considering this plugin. It has been achieved in C language. Implementing it in MATLAB should give better insight into whether or not it is a viable technique since the precision is better.

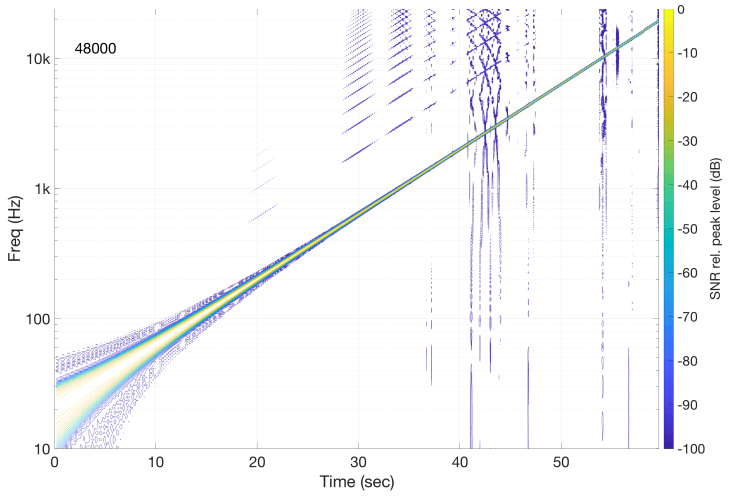


FIGURE 3.26: Quality Evaluation of The Smarc Project

We already know that the first multistage method is performing as desired since the above quality assessements were measured on systems constructed with it. Regarding the second manner, a simple comparison was effected: recreating in MATLAB the implementation of a four-stage folded Parks-McClellan filter in the same way as the plugin (and as [20]). The comparative results are given in the table below.

Points of Comparison	**Multistage Method**	
	Classic Method	Second Method
Specifications (R_p, R_s, TW)	(0.1, 140, 0.85)	(0.1, 140, 0.85)
Best Combination	((4,1), (5,7), (4,7), (2,3))	((4,1), (4,7), (5,7), (2,3))
MPOS	255.9408	172.4075

TABLE 3.3: Comparison of Multistage Methods

This comparison suggests that the second method enables a better efficiency. To see why, we can apply the theory described in section 3.1.4.2 to this pratical case (see figure 3.27).

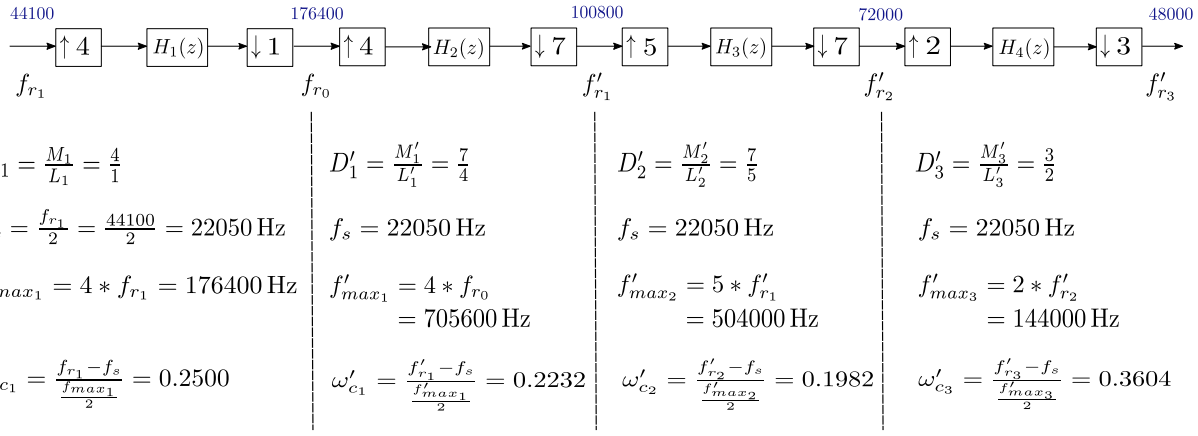


FIGURE 3.27: Application of the Smarc's Method to a Practical Case

If we compare the values of the digital cutoff frequencies ω_{c_i} and ω'_{c_i} to the ones of the classic method (cf. table 3.4), the first ones are definitely larger. Thus the filters designed by the second approach have a flatter roll-off which reduces their complexity (the order).

Stage 1	Stage 2	Stage 3	Stage 3
$\omega_{c_1} = 0.2500$	$\omega_{c_2} = 0.1429$	$\omega_{c_3} = 0.1429$	$\omega_{c_4} = 0.3333$

TABLE 3.4: Cutoff Frequencies of the Classic Method for the Same Practical Case

However, we have to ensure that the quality of the conversion is as satisfactory as the one obtained with the classic method. For the specifications given in the above table, the output takes the following form.

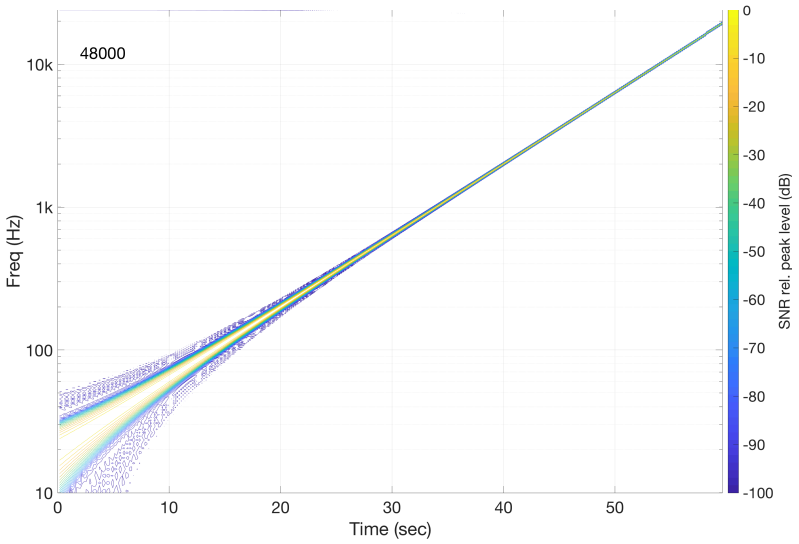


FIGURE 3.28: Sweep Frequency Response Analysys of the Smarc Method

Only a barely visible artifact is displayed on the graph. It is located at the very top of the figure, from 0 to 30 sec. We can still retain this second method since this artifact is very low in decibel, and outside of the range of the human hearing. Moreover, we could slightly increase the filter's specifications to remove it if we absolutely want it.

3.2.7 Final Choice

If we gather all the informations collected so far, we can suggest a final configuration for the conversion from 44100 Hz to 48000 Hz. Keeping in mind that we are not wholly sure about the effect of the non-linear phase of an IIR filter on the audio perception, we decided to give two solutions for this study case. The results are summarized below.

For the conversion from 48 kHz to 44.1 kHz ($M > L$), the filter remains the same according to equation 3.2. On the other hand, the order of the multistage factors should be different. But, considering that the number of stages will be the same and with respect to the conclusion of 3.1.4.2, the process of arranging the stages can be automated.

Features	Filter Structure	
	Parks-McClellan Folded	Elliptic Russell
Polyphase Decomposition	Yes	Yes
Multistage Method	Second	None
Number of Stages	4	1
Specifications (R_p, R_s, TW)	(0.1, 140, 0.85)	(0.1, 140, 0.85)
Best Combination	((4,1), (4,7), (5,7), (2,3))	(160, 147)
MPOS	172.4075	30.7063

TABLE 3.5: Best Configuration

3.2.8 Conclusion

In this chapter, we have presented a way to optimized a synchronous sampling rate conversion through an example. If the choice falls on a FIR Parks-Mclellan filter, a folded four-stage polyphase implementation is the most optimized way. On the other hand, if one decides to use an IIR Elliptic filter, a single stage is enough if we use a particular approach allowing a polyphase decomposition of the FIR part.

3.3 GStreamer

This final part has a vocation to explain the current sample rate converter used by Bang & Olufsen in their products and why this project is important.

3.3.1 What is Gstreamer?

GStreamer is a framework creating streaming multimedia applications involving video such as audio. It is based on the use of plugins written in C to handle different type of processing (format conversion, resampling, ...). The aim is to create a *pipeline* connecting different plugins in order to achieve what we desire. A plugin can be considered as a processing block which can receive input data or *buffers* by means of a *sink pad* and deliver them to the next plugin through an ouput or *source pad*.

In the case of this project, the pipeline is composed of 5 main plugins:

- *GstURIDecodeBin*: itself composed of 3 elements: this plugin goes fetch a file that will be processed. Typically, here it is a WAV file containing an input sweep such as 3.23. Then, it converts the file to a raw format and extracts some useful data such as the rate, the number of channels, etc.
- *GstAudioConvert*: this second plugin converts raw audio buffers between various possible formats. It supports integer to float conversion, width/depth conversion, signedness and endianness (little endian to big endian and vice versa) conversion and channel transformations (ie. upmixing and downmixing), as well as dithering and noise-shaping
- *GstAudioResample*: this is where the actual sampling rate conversion is happening. We will mainly focus on this particular block.
- *GstCapsFilter*: it permits to set *capabilities* or *caps* in other words, describes the type of data that is streamed between two pads, or that one pad supports. Here we set the output type which is raw format and the output rate is 48 kHz.

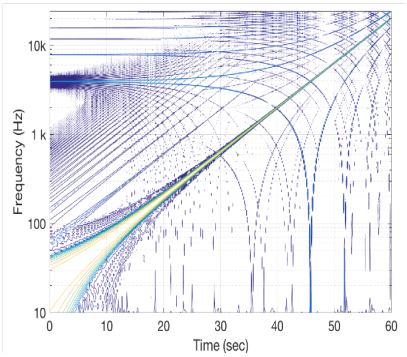
- *GstFileSink*: this last plugin will simply save the raw file on the laptop. The next step would be to convert it back to a WAV file to then treat it and obtain a sweep frequency response analysis.

3.3.2 Core Algorithm

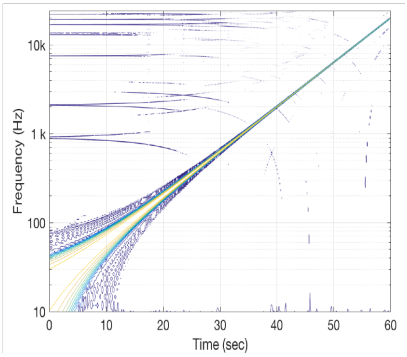
As previously stated, the plugin that we are interested in is the AudioResample one. The algorithm is quite similar to the one described in [4]. By default, the plugin works as follows: several classic Sine Cardinal filters windowed by a *Kaiser* window (see [5]) are used to calculate output values equally spaced between two known input samples. Then, in order to compute an output sample at the time $T_{out} = 1/F_{s_{out}}$ which does not fall onto one of the previous computed values, it must estimate its value based on neighboring samples. In order to do so, the algorithm will take four precomputed filter values: two to the left and two to the right of the desired output position. Then it fits a 3rd order cubic polynomial $P(t)$ through these four values and evaluates $P(t)$ at the output time T_{out}.

3.3.3 Efficiency and Quality Aspects

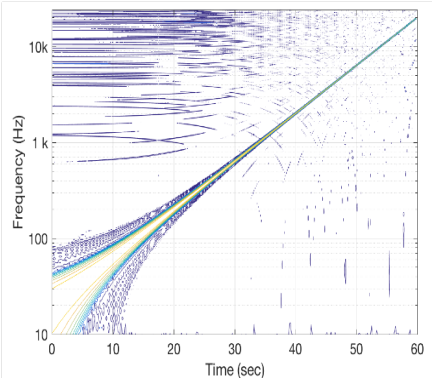
We know from [14] that a filter based on a Kaiser window is not the optimum one but the Parks-McClellan filter is in the FIR category. Moreover, neither polyphase decomposition nor a multistage structure is utilized. Finally, this algorithm is optimal for asynchronous conversion. In terms of quality, the next figure gives an overview of the conversion achieved by the GStreamer plugin.



(a) Quality 0



(b) Quality 4



(c) Quality 10

FIGURE 3.29: Output From GStreamer Plugin

The quality makes a reference to the stopband attenuation R_s, the transition TW bandwidth and the number N of kaiser filters used to compute values between two input samples. The quality varies from 0 to 10 with (R_s, TW, N) = (60, 0.86, 4) to (R_s, TW, N) = (120, 0.975, 32) respectively. Even though we expect a better conversion while the quality parameter is enlarged, it is apparently not the case according to the graphs in the figure 3.29: the outcome is better for a quality parameter of 4 than 10.

3.3.4 Conclusion

Whatever it be, the output from the plugin is extremely poor in all cases. Thus, this project aims to provide a solution to the problem. What remains to be done in the project is to implement a extra method in the GStreamer puglin applying the two concepts retained in 3.2.7. However, some further improvements are still possible.

3.4 Future Work

We detail in this last section some thoughts that might be worth exploring to further enhance the converter.

- Investigate the effect of the ripple in the passband on the auditory perception. Indeed, if the impact is negligible, increasing its value would lower the complexity of the filter even more and then the overall number of MPOS. Keeping in mind that a too important ripple would create a *blip* or pre-echo in the filter's impulse response, thus deteriorating the quality of the conversion. [22] and [23] could be a start.

- There might be a filter implementation faster then the others. In [3], the techniques of block convolution overlap-add/save are said to be highly efficient for FIR impulse responses of even modest length (on the order of 25 or 30), other algorithms could also be considered as (cf. [24]). We could use these algorithms for the non-recursive polyphase components. The recursive part (IIR) can be implemented using the other structures, like a second order sections to avoid potential numerical problems caused by coefficient quantization.

- Continuing the evaluation of the audio quality by confirming that the effect of the phase distortion created by the IIR filters is negligible and that the overall impulse response of the system is the one expected.

Chapter 4

Conclusion

Sampling rate conversion is a broad topic. If we desire to modify two known sample-rates, we can opt for a synchronous conversion. On the other hand, an asychronous conversion would be preferred if the sampling frequencies are unknown or permanently evolving. This report aimed to present manners to optimize a synchronous procedure.

A major challenge was the choice of the low-pass filter and its design. Whether or not the phase distortion is perceptible, the student decided to use two kinds of filter: a FIR Parks-McClellan filter and an IIR Elliptic filter. These two filters were retained because they have the lowest order within their category. Typically, two known sampling frequencies can be related by a ratio L/M. If it is greater than 1, we *upsample* the signal, and *downsample* it when the ratio is lower than 1. The direct method would be to use an *expander* L that would strech out the signal, a low-pass filter to remove replicas previously created and alising, and finally a *decimator* that would keep every M^{th} output samples only. An implementation in this way is often not achievable because of the large values of L and M. Important savings can be reached for the FIR case if we decide to implement the filter in a multistage decomposition and parallelizing its impulse reponse by generating its polyphase components. The IIR case does not require a multistage separation, but by manipulating its transfer function, a FIR part can be extracted and thus implemented as a polyphase decomposition too.

These approaches gave good quality evaluations, especially compared to those obtained by the current process, GStreamer. Nevertheless, certain points remain to be explored and a fully-functioning program must still be incorporated in the current plugin.

Appendix A

Multistage

A.1 MPOS for Some Combinations of a Four-Stage Parks-McClellan

Combination	MPOS
((4,1), (5,7), (4,7), (2,3))	255.9408
((4,1), (8,7), (5,7), (1,3))	283.1575
((10,1), (4,7), (4,7), (1,3))	286.8027
((2,1), (16,7), (5,7), (1,3))	298.0788
((20,1), (2,7), (4,7), (1,3))	300.3986
((40,7), (1,1), (4,7), (1,3))	318.4238
((80,7), (1,1), (2,7), (1,3))	324.1601
((20,1), (8,7), (1,7), (1,3))	357.6583
((16,1), (10,7), (1,7), (1,3))	366.0996
((4,1), (1,1), (40,49), (1,3))	401.5166
((2,1), (80,49), (1,1), (1,3))	420.0163
((2,1), (1,1), (20,21), (4,7))	474.9893
((2,1), (1,1), (16,21), (5,7))	535.2768
((8,1), (1,1), (20,49), (1,3))	549.506
((4,1), (2,1), (20,49), (1,3))	555.1182
((4,1), (1,1), (10,21), (4,7))	591.0457
((10,1), (1,1), (16,49), (1,3))	625.436
((5,1), (2,1), (16,49), (1,3))	629.8442
((5,1), (1,1), (8,21), (4,7))	653.4855
((40,21), (1,1), (1,1), (4,7))	784.7607
((4,1), (4,1), (5,21), (2,7))	835.0173
((4,1), (4,1), (10,49), (1,3))	862.9507
((8,7), (1,1), (1,1), (20,21))	868.5381
((80,21), (1,1), (1,1), (2,7))	869.4054
((10,7), (1,1), (1,1), (16,21))	939.8631
((5,1), (4,1), (8,49), (1,3))	1014.8754
((4,1), (4,1), (5,49), (2,3))	3931.0173

TABLE A.1: MPOS for Some Combinations of a Four-Stage Parks-McClellan

References

[1] P. Martínez-Nuevo, H. Lai, and A. V. Oppenheim. "Amplitude sampling". In: *Allerton Conference on Communication, Control and Computing* (2016).

[2] J. G. Proakis and D. G. Manolakis. *Digital Signal Processing: Principles, Algorithmes and Applications*. Fourth Edition. Prentice Hall, 2009.

[3] A. V. Oppenheim and R. W. Schafer. *Discrete-Time Signal Processing*. Third Edition. Pearson New International Edition, 2013.

[4] P. Beckmann and T. Stilson. "An Efficient Asynchronous Sampling-Rate Conversion Algorithm for Multi-Channel Audio Applications". In: *Audio Engineering Society, 119th Convention* (Oct. 2005).

[5] J. O. Smith III. "The Digital Audio Resampliong Home Page". In: *Center for Computer Research in Music and Acoustics (CCRMA), Stanford University* (2018). URL: https://ccrma.stanford.edu/~jos/resample/.

[6] P. Martínez-Nuevo. "Nonuniform Sampling Rate Conversion: An Efficient Approach". In: *In preparation for IEEE Transactions on Signal Processing* (2018).

[7] J.-G. Chung J.-S. Park B.-K. Kim and K. K. Parhi. "Design of a Sample-Rate Converter From CD to DAT Using Fractional Delay Allpass Filter". In: *IEEE Transactions on Circuits and Systems-II: Express Briefs, Vol. 54, No. 1* (January 2007).

[8] M. Blok. "On Sample Rate Conversion Based on Variable Fractional Delay Filters". In: *International Journal of Computer Science and Applications, Vol. 10, No. 1, pp. 98 - 116* (2013).

[9] R. E. Crochiere and L. R. Rabiner. *Multirate Digital Signal Processing*. Vol. 10. Prentice-Hall Englewood Cliffs, NJ, 1983.

[10] P. P. Vaidyanathan. *Multirate Systems and filter banks*. Pearson Education, India, 1993.

[11] F. E. Toole. *The Acoustics and Psychoacoustics of Loudspeakers and Rooms - The Stereo Past and the Multichannel Future*. 109th AES Conv., Los Angeles, Sept. 2000.

[12] W. M. Hartmann. *Signals, Sound, and Sensation*. Springer, N. Y.

[13] S. P. Lipshitz, M. Pocock and J. Vanderkooy. "On the Audibility of Midrange Phase Distortion in Audio Systems". In: Vol. 10 (Sept. 1982).

[14] P. Martínez-Nuevo. "Filter Design for a Rate-Conversion System".

[15] The Mathworks Co. *Matlab and Simulink*. Natick,MA.

[16] J. S. Lim, A. V. Oppenheim. *Advanced topics in signal processing*. Prentice Hall, 1988.

[17] P. P. Vaidyanathan. *Multirate Systems and Filter Banks*. Prentice Hall, 1993.

[18] A. I. Russell. "Efficient Rational Sampling Rate Alteration Using IIR Filters". In: No. 1 (Jan. 2000).

[19] D. I. Turek. “Design of Efficient Digital Interpolation Filters for Integer Upsampling”. MA thesis. Massachusetts Institute of Technology, Jun. 2004.

[20] R. E. Crochiere and L. R. Rabiner. “Optimum FIR Digital Filter Implementations for Decimation, Interpolation, and Narrow-Band Filtering”. In: No. 5 (Oct. 1975).

[21] Jacques Prado. *Conversion de Fréquence*. Tech. rep. Télécom ParisTech, Sept. 2009.

[22] N. Amir, M. Milhim and L. Radai. “Thresholds for Perception of Ripple in the Passband of a Low-Pass Filter”. In: (Issue 2-3 2011).

[23] A. Ya. Supin, V.V. Popov, O. N. Milekhina, and M. B. Tarakanov. “Ripple Depth and Density Resolution of Rippled Noise”. In: (Issue 5 1999).

[24] J.-G. Chung and K. K. Parhi. “Frequency Spectrum Based Low-Area Low-Power Parallel FIR Filter Design”. In: (2002), pp. 944–953.