Synchronous Sample Rate Converter for Digital Audio

Luís Martins[1]
[1]Instituto Superior Técnico, Av. Rovisco Pais 1, 1049-001 Lisboa

The work carried out intends to find a method to perform the conversion of sampling frequencies for digital audio signals in a simple, straightforward way and that occupies few resources on a FPGA. During this work two implementations were made, one in software and one in hardware. The software implementation aimed to perform tests with different parameters quickly and efficiently. The parameters tested were the fundamental signal frequency, the input sampling frequency, the interpolator polynomial order, and the sample rate ratios. Configurations with and without filtering were also tested. The purpose of the second implementation was to have a prototype that could be deployed on a FPGA and would take up few resources. A direct method (Lagrange polynomial) was used to perform the conversion, because it uses explicit formulas to obtain the coefficients, and there is no need for optimization. Additionally, for low frequencies they feature a very high quality ideal for digital audio. Finally, it can be implemented with efficient algorithms and structures, reducing computational effort. A high polynomial order was used, because it was found that anti-*aliasing* filters were resource intensive, but didn't add significant improvement. The hardware implementation was validated by simulation and the total harmonic distortion plus noise in the Nyquist band was below -90 dB for some predefined sample rate ratios.

***Index Terms*—SRC, Lagrange, FPGA.**

I. Introduction

IN digital signal processing, sample rate converters (SRC) are used whenever you want to convert an input sampling frequency to any other sample rate [1]. A common example in the literature is converting compact disk (CD) to digital audio tape (DAT), where a signal sampled at 44.1 kSa/s is converted to 48 kSa/s [2]. Generally, SRC are required whenever media formats must be converted between systems, as sampling frequencies are typically different and signal reproduction is only possible if the sampling frequency matches the system's sampling frequency that will do the playback of the digital audio.

In the industry various sampling frequencies are used, for example, communication quality on mobile phones use 8 kHz, telecommunication services uses 16 kHz, broadcasting services uses 32 kHz, CD players use 44.1 kHz, household products use either 44.1 kHz and professional applications use 48 kHz [5, 6].

There is a traditional way to approach the problem of converting different sampling frequencies by using a digital-to-analog (DA) converter together with an ideal filter and an analog-to-digital (AD) converter. This solution was used in the past. Successive conversions performed by these converters introduce significant errors and the cost associated with adding this excessive number of converters and filters is also a reason to look for more efficient solutions [7]. With the emergence of digital studios it is common to do all signal processing digitally, for example with recording, editing and production of signals. The techniques currently used involve digital techniques and have evolved to be more efficient and complex.

A. Sampling theorem

To understand this section it is important to consider that from the continuous time signal $x_c(t)$ it is always possible to obtain a discrete time signal $x[n]$. However the opposite conversion may not be possible. The conditions for this conversion to happen are set out in Nyquist's theorem that states if a continuous signal $x_c(t)$ has a limited band in the Fourier transform $X_C(j\Omega)$, ie $|X_C(j\Omega)| = 0$ for $|\Omega| \geq 2\pi F_C$, $x_c(t)$ can be reconstructed without error from equidistant samples $x_c(nT)$, $-\infty < n < \infty$, if $F \geq 2F_C$ where $F = \frac{1}{T}$ is the sampling frequency.

If the sampling theorem is fulfilled, the discrete frequency sampling is given by equation 1:

$$X(\mathrm{e}^{j\omega}) = \frac{1}{T_s} X_C(\frac{jw}{T_s})\,. \tag{1}$$

Where, $X(\mathrm{e}^{j\omega})$ is the discrete Fourier transform of the discrete time signal, ω is the normalized angular frequency, T_s is the sampling period and $X_C(\frac{jw}{T_s})$ the discrete Fourier transform of the continuous time signal.

B. Signal reconstruction

The main idea to keep from the sampling theorem is that any continuous signal can be recovered as long as the sampling frequency meets the minimum requirements. The technique for reconstructing continuous signals from discrete signals is called filtering. Filtering is useful to retrieve the signal spectrum $X_C(j\Omega)$ after convolution $X_C(j\Omega) * S_C(j)\Omega)$. It also requires a DA converter to obtain $x_c(t)\,s(t)$ from $x[n]$. Where $s(t)$ is the periodic impulse train (sampling function).

The reconstruction formula is obtained with equation 2:

$$x_c(t) = \sum_{n=-\infty}^{\infty} x[n]\widehat{h}(t - nT)\,. \tag{2}$$

II. Sample Rate Converters - SRC

When it comes to sampling frequency conversion, it means that the main goal is using the sequence of discrete time samples resulting from sampling a continuous time signal,

which has been sampled at a certain input sampling frequency, and by using some arbitrary technique, create a new sequence of discrete samples. Equivalent to the sampling of the continuous signal by the output sampling frequency. Figure 1 shows typically how this process is done:

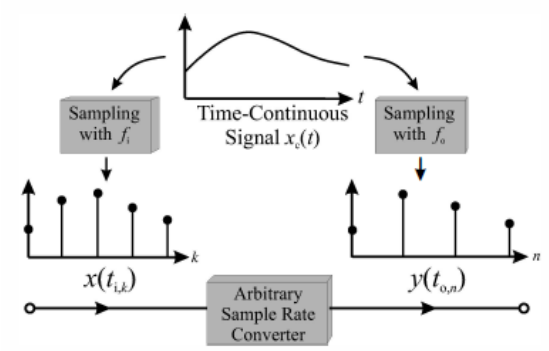


Fig. 1. Conversion model (source [8])

There are two classes of converters: synchronous and asynchronous [7, 9]. Synchronous converters are those with *a priori* known input sampling frequency and conversion factor. In the case of asynchronous converters, as opposed to synchronous converters, the exact sampling frequencies are not known, because the clock signal belongs to different equipments and the time reference of these systems are different.

Conversion techniques can be divided into two types:

- Analog approach;
- Digital approaches.

A. Analog approach

For analog conversion it is common to use a DA converter to reconstruct the continuous signal. Then a low pass filter is used to eliminate replicas, and the resulting signal passes through an AD converter to sample at the desired output frequency.

Due to the $\widehat{h}(t)$ filter, the analog low-pass filter impulse response will have a finite duration and the discrete sequence values $y[m]$ at the time $t = mT'$, T' is the sampling period of $y[m]$, will be obtained through the sample sequence $x[n]$.

$$y[m] = x_c(t)|_{t=mT'} = \sum_{n=N1}^{N2} x[n]\widehat{h}(mT' - nT)\,. \quad (3)$$

$N1$ and $N2$ correspond to the upper and lower limits of the samples for calculating each value of $y[m]$.

The discrete sequence $x[n]$ samples and weights the impulsive response. The summation limits of the equation 3 are determined where the ideal filter is nonzero. Assuming that $\widehat{h}(t)$ is null for $t < t_1$ and $t > t_2$, one can derive the equations 4 and 5. These equations show the relationships between input (T), output (T'), sample m, and lowpass filter boundaries t_1 and t_2.

$$N1 = \lceil \frac{mT' - t_2}{T} \rceil\,. \quad (4)$$

$$N2 = \lfloor \frac{mT' - t_1}{T} \rfloor\,. \quad (5)$$

There are two peculiar situations: The first situation is when you have two impulsive responses of different length for the same conversion, if the indexes are even, the $x[n]$ sample set that forms $y[m]$ is the same, but if the indices are odd then the $x[n]$ sample set that forms $y[m]$ is different. The second situation involves low pass filter coefficients that vary depending on the ratio you want to convert [10].

If we consider that there is a relationship between sampling frequencies rather than being arbitrary, we have:

$$\frac{T'}{T} = \frac{M}{L}\,,. \quad (6)$$

Then, there are L unique coefficient sets for the analog low pass filter that are used to calculate $y[m]$ from the discrete sequence $x[n]$. If the ratio were irrational there would be infinite sets of coefficients.

B. Digital conversion techniques

It is desired to avoid analog conversion, ie the goal is to perform conversions purely in the digital domain.

1) Direct Method

One method to do it is the direct method. To implement it proceed as follows: first, the impulsive response of the analog filter is interpolated L times the input sampling frequency. The set of L samples will correspond to the unit response of the digital filter and these samples are also used as coefficients to calculate output samples $y[m]$, reminding that there are L sets of unique coefficients. Two very important properties of these systems is their linearity and periodicity [10].

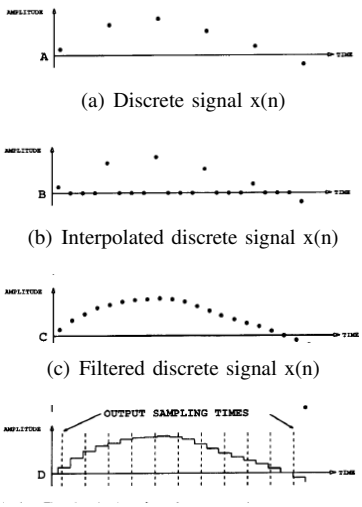
2) Decimation

The process that results from reducing the sampling frequency is called decimation. The easiest way to do this is to save one sample every M samples and ignore the rest, with the new sampling frequency being: $f_{new} = \frac{f_{old}}{M}$. The signal spectrum will shift around f_{new} and all its multiples. Care must be taken as the input signal band must be limited, otherwise it may occur *aliasing* and information is lost, so the condition must be imposed: $f_{new} > 2B$, where B represents the signal band. In some situations you may need to use a low pass filter before decimating, for example if you need to use a sampling frequency of less than $2B$, then you need to filter first with a low pass filter and only then perform decimation [11].

3) Interpolation

Figure 2 shows the various temporal steps of a sampling frequency converter. Figure 2 a) is the discrete input signal to be converted, then zeroes are added to the discrete signal to increase its sampling frequency as shown in figure 2 b). However, this signal needs low pass filtering to obtain the new values (interpolation) corresponding to the continuous signal figure (2 c). Next, the discrete signal (on a finer time scale) is converted to a continuous signal figure (2 d). Generally after reconstruction, the output signal is sampled asynchronously and the values obtained at the output correspond to the closest time points to the values produced by the interpolator filter. There is always an error, as the sample retention switch does not close exactly at the ideal time, ie the output samples are

slightly spaced from their position relative to the thinnest time scale. The solution to this problem is to increase the number of samples by increasing the interpolation factor because it creates a smaller time spacing and adjacent samples get closer [7].



(a) Discrete signal x(n)

(b) Interpolated discrete signal x(n)

(c) Filtered discrete signal x(n)

(d) Continuous signal $x_c(t)$

Fig. 2. Interpolation in the time domain (source [7]).

4) Conversion by a rational factor L/M

So far it has been found how to increase the sampling frequency by an integer factor L and how to decrease by an integer factor M. However, there are situations where the conversion by a non integer factor is needed. In these scenarios, a rational L/M conversion is applied, where the sampling frequency is first interpolated by an integer factor L, then passes an anti-image lowpass filter, then proceeds to an anti-*aliasing* lowpass filter, and then decimated by an integer factor M. Since during digital processing of the discrete value sequence both filters are operated at the same sampling frequency, they can be merged into a single low-pass filter that acts as an anti-image and anti-aliasing filter. Interpolation always precedes decimation to maximize the bandwidth of the input signal. However, it is only a theoretical model since it is quite inefficient.

5) Fractional delay filtering

A fractional delay filter is a device that delays a discrete signal by a fraction of a sample [12, 13]. A commonly used structure for interpolation filters is based on fractional delay FIR filters. Can be used FIR or IIR *Infinite Impulse Response* filter structures to implement fractional delay filters [13]. FIR filter structures will be addressed, essentially because they are simpler to design and analyze. The idea of fractional delay filters is explained in [14]. Basically the limited band continuous signal is rebuilt and after moving it, resampling is performed. A fractional delay filter transforms a discrete sequence of time values $x[n]$ into a sequence of discrete time values $y[n, D]$, which approximates $x[n]$ delayed by a time delay D given in samples [15]. The D parameter is not usually an integer. To control the time frames for output samples, it is necessary to synthesize several FIR filters with various delays and the coefficients must be stored in a *look up table* - (LUT). The desired time point for the interpolated output sample can be controlled by weighing the FIR filter output samples by their fractional interval [16].

6) Asynchronous frequency conversion

The alternative way of converting sampling frequencies is to transform the input sampling frequency into an output sampling frequency using frequency compressors and expanders. With this technique, the main precautions involve attenuation of signal images as well as aliasing. Asynchronous converters are more prominent if the ratio is formed by two very large integers, if the ratio changes over time, or if the ratio is not formed by an integer ratio.

III. Validation of the converter quality

The total harmonic distortion plus noise (THD + N) was the metric chosen to evaluate system performance as it is the most commonly used metric to measure distortion of audio signals. Measures noise and distortion added to a signal by any equipment. It is determined by applying a single sinusoidal signal (with known harmonic content) to the input of the test circuit, and the distortion is determined. Then the frequency of the signal used in the test is filtered by a notch filter, passing the resulting signal through a series of filters adjusted to the band of interest (20 Hz - 20 kHz). What remains is noise and the harmonics generated by the equipment. The definition used for THD + N is in the equation 7.

$$\mathrm{THD + N} = \frac{\sum_{n=2}^{\infty} \mathrm{harmonics + noise}}{\mathrm{fundamental}} \,. \tag{7}$$

IV. Proposed solution

The proposed solution is simple and is based on Lagrange interpolating filters with an FIR filter structure. The implementation is purely digital and uses the direct method.

Lagrange polynomial is a form of polynomial interpolation used for numerical analysis. For a given set of points (x_k, y_k) with no two x_k values equal, a K-order polynomial is obtained. Which is the polynomial of lowest degree that makes the correspondence between the x_k and y_k) values. This polynomial interpolates arbitrary points in the range $[x_0, x_K]$. Lagrange interpolation is a polynomial interpolation class, based on Lagrange [15] polynomials.

The Lagrange interpolator is obtained by linear combination of Lagrange polynomials, as depicted in equation 8.

$$L(x) := \sum_{j=0}^{k} y_j \, l_j(x) \,. \tag{8}$$

Lagrange polynomials are obtained with equation 9.

$$l_j(x) := \prod_{i=0, j\neq i}^{k} \frac{(x - x_i)}{(x_j - x_i)} \,. \tag{9}$$

Properties with practical interest are:

- The polynomial coefficients can be calculated in closed form (with explicit formulas) and therefore no optimization is needed [20];

- Low-order Lagrange interpolation polynomials are efficient with the hardware they use and perform well with high attenuation for narrowband signals (ie a band away from Nyquist frequency) [13, 19].
  At low frequencies these polynomials are of very high quality, making them an ideal choice for digital audio applications [15];
- Lagrange interpolator polynomial can be implemented with efficient algorithms and structures, reducing the computational effort of the system [17];
- It is simple to design the filter and also offers several design options for rational sampling frequency converters;
- Using a two-stage structure (Lagrange over-sampling) improves image attenuation and quality for broadband signals [17, 18].

The disadvantages appear in the form of:

- If the signal band is too wide (a band close to the minimum Nyquist frequency) the quality of Lagrange's interpolator polynomial begins to deteriorate [13, 19];
- Frequency response is fixed and only depends on one parameter, the order of the interpolator filter. Optimization is not possible [20];
- The structure (oversampling + Lagrange) are individually designed causing a performance loss [17].

The coefficients can be calculated in two ways: real-time or precalculated [21, 22]. The first approach requires a lot of computational effort to get the coefficients, while the second approach needs a memory to store the coefficients, and also has the disadvantage that it can only make conversions for the stored ratios.

We opted for the second option to save resources on the device. The procedure used was as follows:

1) The coefficients were pre-calculated for some ratios. This calculation was made using a time domain Lagrange polynomial interpolation algorithm using a C program;
2) The impulsive response of the projected analog filter yielded N coefficients corresponding to the impulsive response of the digital filter used for the 'real' interpolation of the sampling frequency converter.
   The N value corresponds to the only configurable parameter of this implementation, which is the order of the Lagrange interpolator polynomial;
3) For the sampling frequency ratio $\frac{L}{M}$ there are L unique sets defined by the interpolation factor. Thus, L unique set of coefficients formed by N coefficients for the conversion ratios (0.4; 0.5; 0.6666; 0.75; 2; 3; 4) was stored in a ROM.
4) To perform the decimation, an offset of M was applied to the memory position every L iterations.
5) For a conversion ratio designed with this implementation, the complexity is on the way coefficients are addressed, and ideally these ratios should be given by small mutually prime integers, to reduce the number of coefficients stored in memory.

V. Software implementation

This chapter explains the program developed in software. It was created because a tool was needed that could allow a quick implementation of Lagrange's interpolation polynomial to test some parameters for its later implementation in hardware. Those tests allowed to see how the interpolator performance was affected. It also served to test some configurations and to understand more generally how the polynomial behaved, for example by varying the order or by using different sample rate ratios and sampling frequencies. Additionally, the program made it possible to create a comparison reference for the conversion performed in hardware, serving as a validation of the desired functionality. The program was designed in fixed point arithmetic to match the hardware implementation.

A. Algorithm description

The code that was performed is divided into two programs. The first program $sinewave.c$ is responsible for creating the input samples, obtained by sampling a sinusoid for a given input frequency. It is also this program that creates the reference, ie the expected result after conversion.

The second program $lagrange.c$ implements the proposed solution. This program operates with fractional or integer sampling frequency conversion ratios. It also allows to change the order of the polynomial. The input values come from the terminal's $stdin$ and are generated by the $sinewave.c$ program. The arithmetic is in fixed comma form to match the later implementation in FPGA.

An external program running on $Octave$ is used to view the reconstructed signal waveforms and to determine the THD + N of that signal. In this work the term points and samples are used interchangeably, as well as the term Lagrange interpolator polynomial and interpolator filter. Note that this program, to calculate the power of noise, discards the DC component and the fundamental harmonic, as well as a frequency window around these two frequencies.

VI. Hardware implementation

This chapter describes the Verilog implementation of the sample rate conversion algorithm, justifying the choices made regarding hardware implementation.

After describing and understanding the sample rate conversion system, it presentes the limitations of the system. For example, the constraints imposed on the $\frac{L}{M}$ ratio to have periodicity in the coefficients and the inability to calculate them in real time due to frequency requirements.

The block diagrams are presented, and the resource usage report occupied by the FPGA implementation is presented.

A. Proposed solution

The circuit developed and synthesized in FPGA has essentially 4 modules each with specific and distinct functions. Starting with memories: A read-only ROM memory where the interpolator filter coefficients are stored for the following conversion ratios: 0.4; 0.5; 0.6666; 0.75; 2; 3; 4, which may alternatively be represented as an irreducible fraction

$(\frac{2}{5}; \frac{1}{2}; \frac{2}{3}; \frac{3}{4}; \frac{2}{1}; \frac{3}{1}; \frac{4}{1})$. In this form the interpolation factors (L) are obtained directly through the numerator (2, 1, 2, 3, 2, 3, 4), not forgetting that the interpolation factors correspond to the multiplicity of unique sets of coefficients. The way coefficients are organized internally in ROM is in ascending order of conversion ratio. The figure 3 shows the module used, for the ROM memory:

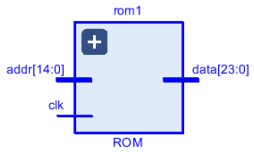


Fig. 3. Module ROM.

The other memory is a RAM memory and allows to store the input samples. The memory used in this implementation uses two clock signals:

- A $clka$ clock signal running at input sampling frequency to write the new samples;
- A $clkb$ clock signal operating at the frequency obtained through the product between the interpolator filter order and the input sampling frequency, which is used for data processing.

The figure 4 shows the module used, for the RAM memory:

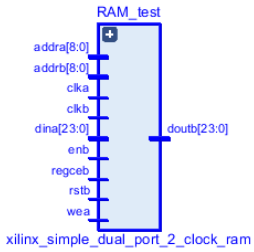


Fig. 4. Module RAM.

Then, there is the finite state machine ($FSM.v$) responsible for controlling the data. This state machine has 3 states: `IDLE`, `READ` and `FULL`. The state machine serves to generate RAM and ROM addresses and to provide control signals to the circuit. It stays in the state `IDLE` whenever RAM has insufficient data to perform the estimation. When the amount of samples stored in RAM is satisfactory, it proceeds to the `FULL` state. This state indicates that all possible output samples have already been estimated, from the input samples stored in RAM. To change state it is needed to save more samples in RAM. It enters the `READ` state when this condition is met. While remaining in the `READ` state, it means that the process of estimating output samples is not yet complete. While in this state it reads coefficients from the ROM and input samples from the RAM, realizing the product of these values. The result of this operation is added to an accumulator. The final value of this accumulator corresponds to the value of the output sample being estimated.

Figure 5 shows the module used for the finite state machine.

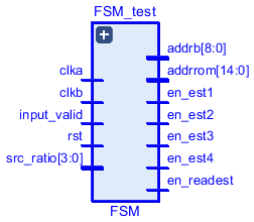


Fig. 5. Module FSM.

The last module ($Circuit.v$) joins the RAM and ROM memories with the data control module ($FSM.v$), forming the sampling frequency conversion circuit. The $Circuit.v$ module addresses RAM for writing input samples. This module uses the control signals provided by the state machine to perform the estimation of the output samples.

In figure 6, the simplified block diagram of the circuit implementing SRC is presented based on the modules described above.

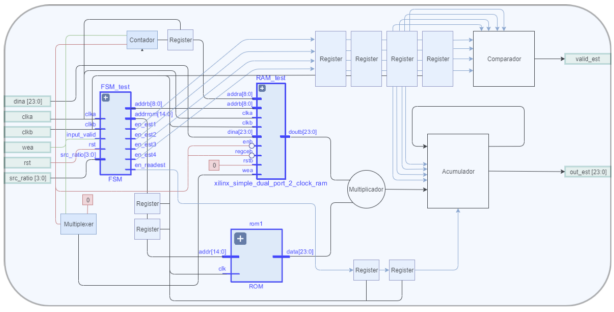


Fig. 6. Block diagram for Circuito.v.

B. FPGA implementation

Figure 7 shows the amount of resources used for the "XC7A12T" device. This FPGA is the smallest of the "Artix-7" family. Figure 8 shows the same results but in percentage form. Therefore, based on the resources indicated, the goal of using a hardware implementation that uses low resources was achieved.

Resource	Utilization	Available	Utilization %
LUT	834	8000	10.43
LUTRAM	4	5000	0.08
FF	263	16000	1.64
BRAM	12.50	20	62.50
DSP	2	40	5.00
IO	57	112	50.89

Fig. 7. Resources used in the FPGA.

Circuit latency was calculated. The first iteration needs 516 cycles of the $clkb$ clock signal, the remaining iterations need only 512 clock cycles. This number is consistent with the amount of samples and coefficients that must be read from RAM and ROM.

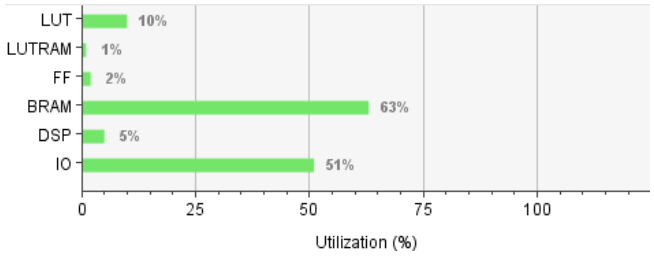


Fig. 8. Percentage resources used in the FPGA.

VII. Results

Before presenting the experimental results it is convenient to define the minimum theoretical value that the THD + N can reach. There is a -144 dB limit due to 24-bit quantization. When calculating THD + N this error caused by quantization is accounted for as noise.

Starting at a very early stage in the implementation where only integer sample rate conversion greater than 1 were allowed, input signal frequencies were sweeped for various interpolation filter orders, various input sampling frequencies and various conversion ratios. Figure 9 shows some settings for typical sampling frequencies (44.1 kHz and 48 kHz) and conversion ratios 2 and 4.

	FS in	48,000					
	ratio	2					
	FS out	96,000					
	order of polynomial	16	32	64	128	256	512
signal freq	1000	-125.7	-120.79	-116.24	-111.55	-106.36	-100.99
	5000	-125.93	-120.72	-116.48	-111.49	-106.39	-100.99
	10000	-87.398	-120.4	-116.42	-111.8	-106.68	-101.32
	15000	-41.505	-69.791	-115.57	-111.95	-106.89	-101.48
	20000	-15.076	-22.508	-34.768	-56.624	-97.445	-101.98

(a) Sweep for sampling frequency of 48kHz and sample rate 2

	FS in	44,100					
	ratio	4					
	FS out	176,400					
	order of polynomial	16	32	64	128	256	512
signal freq	1000	-125.15	-119.98	-115.17	-110.32	-105.12	-99.748
	5000	-125.4	-120.01	-115.29	-110.33	-105.12	-99.749
	10000	-79.112	-119.8	-115.21	-110.25	-105.1	-99.754
	15000	-34.409	-55.297	-94.529	-110.87	-105.7	-100.37
	20000	-19.033	-22.564	-27.924	-36.466	-50.961	-91.08

(b) Sweep for sampling frequency of 44.1kHz and sample rate 4

Fig. 9. Early results.

The analysis of figure 9 shows that as the signal frequency approaches Nyquist frequency, $\frac{F_s}{2}$, the effects of aliasing and therefore THD + N increases. On the other hand, for low signal frequencies (eg 1 kHz) it is noted that increasing the filter order leads to worse results. This is because Lagrange's polynomial produces unpredictable results between samples, which are most noticeable when the order of the polynomial is high, as it can have large oscillations between consecutive samples.

At an early stage, we sought to improve the quality of the results by using an anti-aliasing filter, implemented with a FIR filter. To this end, two programs have been developed in $Mathematica$, one to check the effects of filtering on an interpolated signal and the other to calculate FIR low-pass filter coefficients. Usually to avoid aliasing the filter should cut into $\frac{F_s}{2}$, ie π, but as the filter is not ideal, there will be a little aliasing. By setting a given minimum input frequency (eg 44.1 kHz) then it is possible to cut a little earlier. We tried to manipulate the passband and transition bands in order to have better THD + N in the generated output samples. Figure 10 shows the THD + N obtained by using anti-aliasing FIR digital filters. At this stage of implementation it was already possible to choose any conversion ratio $\frac{L}{M}$.

SEM FILTRO			order of polynomial	
FS in	FS out	src_ratio	256	512
44,100	96,000	2.176871	-39.826	-66.3
44,100	48,000	1.088435	-39.978	-68.528
signal freq	20,000			
Number of Periods	250			

(a) Reference table.

			order of polynomial	
FS in	FS out	src_ratio	256	512
44,100	96,000	2.176871	-61.96	-89.085
44,100	48,000	1.088435	-2.7994	-30.062
signal freq	20,000			
Number of Periods	250			
a=EquiripplefilterKernel[{{{0,0.46 Pi},{0.46 Pi+0.04,Pi}},{1,0}},128]				

(b) Configuration 1

			order of polynomial	
FS in	FS out	src_ratio	256	512
44,100	96,000	2.176871	-61.385	-88.356
44,100	48,000	1.088435	-38.338	-66.298
signal freq	20,000			
Number of Periods	250			
a=EquiripplefilterKernel[{{{0,0.45 Pi},{0.45 Pi+0.05,Pi}},{1,0}},128]				

(c) Configuration 2

Fig. 10. THD + N for FIR filters with anti-aliasing properties.

It can be seen that configuration 2 improves the THD + N against the reference (figure 10 (a)) relative to src_ratio equal to 2.176871. However it slightly worsens for src_ratio equal to 1.088435. These ratios were used because at the time, they were the only ratios which had a THD + N greater than -90 dB, which was the requirement to achieve good audio quality. Regarding the *EquiripplefilterKernel* functions, they are used to create a filter with equal oscillations in the bands. The parameters it receives are the passband location, the attenuation band location, the values in those bands and the filter order.

It was decided not to use the anti-aliasing filter and to use a higher order Lagrange polinomial. The filter allowed to lower the polynomial order, but implied more hardware in its implementation. Eliminating the filter, it means that it is only needed the hardware to implement Lagrange's polynomial and a slightly larger memory to store more coefficients.

The software implementation yielded the results shown in figure 11.

From the analysis of the figure, it can be seen that regardless of the signal frequency, for the conversion ratios 0,5 and 2, the THD + N are always lower than -90 dB if the polynomial order is greater than or equal to 256. For the signal frequency of 1 kHz, excluding these two conversion ratios, no conversion ratio has been able to reach the -90 dB level and the THD + N values are close to -76 dB. For the 20 kHz signal frequency the THD + N clearly improves with the order of the polynomial. We have chosen a polynomial order equal to 512, because a compromise is reached between quantity of stored coefficients and conversion quality. With this signal frequency and polynomial order, except for the conversion ratio 0.5 and

SEM FILTRO (PRODUTO)			order of polynomial				
FS in	FS out	src_ratio	128	256	512	768	1024
44,100	96,000	2.176871	-76.65	-76.64	-76.63	-76.63	-76.62
48,000	96,000	2.000000	-111.55	-106.35	-100.98	-97.82	-96.10
44,100	48,000	1.088435	-76.64	-76.63	-76.62	-76.62	-76.62
48,000	44,100	0.918750	-76.07	-76.07	-76.07	-76.07	-76.07
96,000	48,000	0.500000	-143.84	-143.85	-143.84	-143.86	-143.85
96,000	44,100	0.459375	-76.02	-76.02	-76.02	-76.02	-76.02
number periods			signal freq				
250			1,000				

(a) Frequency signal of 1 kHz.

			order of polynomial				
FS in	FS out	src_ratio	128	256	512	768	1024
44,100	96,000	2.176871	-25.48	-39.81	-66.19	-80.68	-81.08
48,000	96,000	2.000000	-56.62	-97.46	-102.00	-98.77	-98.28
44,100	48,000	1.088435	-25.69	-39.87	-66.45	-80.77	-81.11
48,000	44,100	0.918750	-56.62	-80.05	-80.14	-80.15	-80.29
96,000	48,000	0.500000	-128.58	-128.58	-128.58	-128.18	-129.05
96,000	44,100	0.459375	-80.14	-80.14	-80.14	-80.14	-80.14
number periods			signal freq				
500			20,000				

(b) Frequency signal of 20 kHz.

Fig. 11. Final results.

2, the conversion ratios 2.176871 and 1.088435 fall within the range of -66 dB and the conversion ratios 0.918750 and 0, 459375 are in the range of -80 dB. The causes for poor quality can be attributed to the oscillatory nature of a very high order polynomial approximation. This effect is called the Runge phenomenon. In cases where it is equal to 0.5 and 2 there are points that coincide, hence the quality increases.

It only remains to get the THD + N results from the samples estimated by the hardware implementation and compare them with the software implementation. Figure 12 show various results of the THD + N analysis for different conversion rates, different input signal frequencies and different polynomial orders.

Hardware implementation results are better than software implementation results. This occurs noticeably for $src\_ratio = 0.6666\ldots$ As it is a repeating decimal, it may happen that the hardware implementation handles this case better due to rounding. The other minor deviations between software and hardware are due to inaccuracies in the calculation of THD + N. In figure 12 the cells are empty for the conversion ratio of 0.4 and signal frequency of 20 kHz, because the Nyquist frequency is being violated. The most notable case occurs for the 0.6666 ...conversion ratio where there is a difference of -42 dB for a signal frequency of 1 kHz and a polynomial order of 512.

Regarding the increase in the order of the polynomial from 512 to 768, there is no significant improvement, and there are even cases where there is a slight increase of THD + N.

In conclusion, it is noteworthy that the hardware implementation achieved a THD + N always lower than -95 dB, which is enough for many real applications.

VIII. Conclusion

The objective of this work was to implement a sample rate converter that uses low resources and has a maximum THD + N of -90 dB.

To fulfill the proposed objective, two descriptions of the algorithm were created. Initially a C program was created to adjust the project parameters and find the best settings. Later, a description was developed in Verilog to carry out the

			Código	
FS in	FS out	src_ratio	Software (C)	Hardware (Verilog)
48,000	19,200	0.4	-101.03	-101.04
48,000	24,000	0.5	-141.16	-141.16
48,000	32,000	0.6667	-79.47	-121.65
48,000	36,000	0.75	-98.57	-98.57
48,000	96,000	2	-101.05	-101.05
48,000	144,000	3	-98.54	-98.54
48,000	192,000	4	-100.02	-100.02
number periods			signal freq	order of polynomial
250			1,000	512

(a) Frequency signal of 1 kHz and polynomial order of 512.

			Código	
FS in	FS out	src_ratio	Software (C)	Hardware (Verilog)
48,000	19,200	0.4		
48,000	24,000	0.5	-122.48	-122.48
48,000	32,000	0.6667	-80.18	-98.32
48,000	36,000	0.75	-99.62	-99.63
48,000	96,000	2	-102.03	-102.03
48,000	144,000	3	-99.50	-99.50
48,000	192,000	4	-101.07	-101.07
number periods			signal freq	order of polynomial
500			20,000	512

(b) Frequency signal of 20 kHz and polynomial order of 512.

			Código	
FS in	FS out	src_ratio	Software (C)	Hardware (Verilog)
48,000	19,200	0.4	-97.86	-98.04
48,000	24,000	0.5	-141.15	-141.15
48,000	32,000	0.6667	-79.50	-116.51
48,000	36,000	0.75	-95.42	-95.57
48,000	96,000	2	-97.88	-98.06
48,000	144,000	3	-95.39	-95.57
48,000	192,000	4	-96.92	-97.10
number periods			signal freq	order of polynomial
250			1,000	768

(c) Frequency signal of 1 kHz and polynomial order of 768.

			Código	
FS in	FS out	src_ratio	Software (C)	Hardware (Verilog)
48,000	19,200	0.4		
48,000	24,000	0.5	-122.97	-122.97
48,000	32,000	0.6667	-80.43	-98.74
48,000	36,000	0.75	-96.52	-96.57
48,000	96,000	2	-98.82	-99.05
48,000	144,000	3	-96.35	-96.57
48,000	192,000	4	-97.85	-98.07
number periods			signal freq	order of polynomial
500			20,000	768

(d) Frequency signal of 20 kHz and polynomial order of 768.

Fig. 12. Results from hardware implementation.

implementation. In the Verilog implementation four modules were used: one ROM, one RAM, one FSM state machine and the top circuit.

The results obtained for the software implementation show that there are some rounding errors. In all other cases, it always meets the requirements of the maximum THD + N of -90 dB . For $src\_ratio = 0.5$ it reaches -141 dB, close to the minimum theoretical limit of the THD + N which is -144 dB.

The results obtained for the hardware implementation show that the maximum THD + N is always reached, being the maximum value of -95 dB. The minimum THD + N obtained with this implementation was -141 dB, which is close to the minimum theoretical limit.

One of the disadvantages of using Lagrange's interpolation method is that the coefficients have to be precalculated and, depending on the $\frac{L}{M}$ conversion ratios, it may be necessary to use a ROM with large capacity.

Overall, it was shown that the resources occupied in FPGA

are reduced, fulfilling the objective.

As future work it would be interesting to try another interpolation method, for example B-spline. If Lagrange interpolation is to be maintained, a polyphase structure for the FIR filter may be used, which could lead to some improvements. Another technique to explore would be oversampling, as this is a recurrently used technique.

Acknowledgment

I would especially like to thank my supervisor Fernando Gonçalves, was always available to share both his enormous wisdom on the subject and to provide advice for the thesis to proceed peacefully. I would also like to thank my parents for making sure I finished the course. Finally, I would like to thank my friends who have accompanied me throughout this time.

References

[1] Ronald W. Schafer and Lawrence R. Rabiner, *A Digital Signal Processing Approach to Interpolation*, Procedings of the IEEE, 1973, vol. 61, n. 6, p. 692–702, June.

[2] Marek Blok, 2013, january, p. 98-116, *On sample rate conversion based on variable fractional delay filters*, vol. 10, *International Journal of Computer Science and Applications*.

[3] Jussi Vesma, *Frequency-Domain Approach to Polynomial-Based Interpolation and the Farrow Structure*, *IEEE Transactions on Circuits and Systems — II: Analog and Digital Signal Processing*, 2000, vol.47, n. 3, p. 206–209, March.

[4] Gennaro Evangelista, 2003, February, p. 377-387, *Design of digital systems for arbitrary sampling rate conversion*, vol. 83, *Signal Processing*, doi 10.1016/S0165-1684(02)00421-8.

[5] S. Park and G. Hillman and R. Robles, *ICASSP 91: 1991 International Conference on Acoustics, Speech, and Signal Processing*, *A novel structure for real-time digital sample-rate converters with finite precision error analysis*, 1991, p. 3613-3616 vol.5, April.

[6] R. Lagadec and D. Pelloni and D. Weiss, *ICASSP '82. IEEE International Conference on Acoustics, Speech, and Signal Processing*, *A 2-channel, 16-bit digital sampling frequency converter for professional digital audio*, 1982, vol. 7, p. 93-96, May.

[7] Robert Adams and Tom Kwan, *IEICE TRANSACTIONS on Electronics*, *A Stereo Asynchronous Digital Sample-Rate Converter for Digital Audio*, 1994, vol. E77-C, n. 5, p. 811-818, May.

[8] Ivar Løkken, *The ups and downs of arbitrary sample rate conversion*, d. 12/4-05.

[9] Chunduri SreenivasaRao, A. AroakiaRaj and Dhulipalla VenkataRao, *International Journal of Signal Processing, Image Processing and Pattern Recognition*, *Synchronous Sampling Rate Conversion using Frequency domain based techniques*, 2013, vol. 6, n. 4, p. 13–28, August.

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

[11] Marian Forster, *Sample Rate Conversion in Digital Signal Processors*, Graz University of Technology, 2014.

[12] Ruth Mariela Aguilar Ponce and Gordana Jovanovic-Dolecek, *A multirate approach to design of fractional delay filters*, 1999.

[13] Andreas Franck, Karlheinz Brandenburg, Ulf Richter, *Audio Engineering Society Convention Paper Presented at the 125th Convention*, *Efficient Delay Interpolation for Wave Field Synthesis*, 2008, p. 1–17, October.

[14] T. I. Laakso and V. Valimaki and M. Karjalainen and U. K. Laine, *IEEE Signal Processing Magazine*, *Splitting the unit delay [FIR/all pass filters design]*, 1996, vol. 13, n. 1, p. 30-60, January.

[15] Andreas Franck, *Efficient Algorithms for Arbitrary Sample Rate Conversion with Application to Wave Field Synthesis*, 2012.

[16] J. Vesma and T. Saramaki, *Circuits Systems Signal Processing*, *Polynomial-based interpolation filters—part I: Filter synthesis*, 2007, vol. 26, n. 2, p. 115–146, April.

[17] A. Franck and K. Brandenburg, *2009 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, An overall optimization method for arbitrary sample rate converters based on integer rate SRC and lagrange interpolation*, 2009, p. 301-304, October.

[18] A. Franck, *2011 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, *Arbitrary Sample Rate Conversion With Resampling Filters Optimized For Combination With Oversampling*, 2011, p. 149-152, October.

[19] A. Tarczynski and W. Kozinski and G. D. Cain, *Proceedings of ICASSP '94. IEEE International Conference on Acoustics, Speech and Signal Processing*, *Sampling rate conversion using fractional-sample delay*, 1994, vol. iii, p. III/285-III/288 vol.3, April.

[20] Jaakko Ketola, Jouko Vankka and Kari Halonen, *Synchronization Of Fractional Interval Counters In Non-Integer Ratio Sample Rate Converters*, 2003.

[21] Lars Erup, Floyd M. Gardner and Robert A. Harris, *IEEE TRANSACTIONS ON COMMUNICATIONS*, *Interpolation in Digital Modems-Part 11: Implementation and Performance*, 1993, vol. 41, n. 6, p. 998-1008, June.

[22] Tor A. Ramstad, *Sample Rate Conversion By Arbitrary Ratios*, 1982.