Asynchronous Audio Sample Rate Converter

Pedro Miguel Portela Teixeira
pedro.m.p.teixeira@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

September 2021

Abstract

This thesis proposes a new design of a new 24-bit or less multi-channel Asyncrhonous Sample Rate Converter, with the following specification: (1) total harmonic distortion plus noise ratio (THD+N) of −130 dB or less; (2) supported conversion ratios from 1:24 to 24:1; hundreds of audio channels; (3) synchronisation time of less than 200 ms; (4) variation of the phase between input and output of less than one output sample after a reset; (5) resource consumption per channel similar to the CWda52, or less. All objectives have been met, and hundreds of tests have been applied to the new design, which achieves a THD+N from a minimum of −143 dB to a maximum of −136 dB for a total of 121 tests, with an average of approximately −140 dB. The new design supports conversion ratios range from 1:24 to 24:1. The best alternative solution is the Texas Instruments SRC4194 chip, which supports a range from to 1:16 to 16:1. It can support tens or hundreds of channels using time division multiplexing; the other solutions typically support eight channels, and the CWda52, the main competitor to this approach, needs to replicate its stereo unit to handle more channels. Consequently, the resource usage per channel of this approach is extremely competitive. The synchronisation time is 20 ms, which is one order of magnitude lower than any other alternative. The variation of the phase after reset, an important characteristic omitted in the alternative solutions, is less than one output sample.

Keywords: Asynchronous Sample Rate Converter, Digital Signal Processing, Digital Filter, System-on-a-chip, FPGA

1. Introduction

In 1977, with the growing popularity of audio interfaces, the Audio Engineering Society (AES) founded the AES Digital Audio Standards Committee, creating some of the most used audio standards to this day. One of the most popular standards, the AES3 digital audio interface is still used in current audio equipment, like microphones and speakers with XLR connectors.

Since then, there has been a significant rise in AES standards, many of them demanding the conversion of an audio signal's sample rate. One of the classic examples is the conversion from CD quality with a sampling rate $F_s = 44.1kHz$ to DVD quality with $F_s = 48kHz$. Consequently, there is a great demand for sample rate converters, both in software algorithms and hardware designs.

To answer this need, some integrated circuit manufacturers developed multiple sample rate converters with varied specifications. The AD1896 [3], for instance, is an asynchronous sample rate converter made by Analog Devices, in 2003. It supports a stereo (2 channel) audio signal and converts its sampling rate from 7.75:1 to 1:8 ratios, with a Total Harmonic Distortion Plus Noise (THD+N) ratio of around −125 dB. Another example is the SRC4194 [5], manufactured by Texas Instruments since 2004, supporting up to 4 channel inputs and a wider sampling rate ratio, ranging from 16:1 to 1:16 ratios, with a THD+N of around −140 dB.

The first IP (IP) was developed by Coreworks, an IP design company, and called the CWda52 [7]. It supports up to 8 channels and converts sampling rates between 8 kHz and 192 kHz, with ratios from 7:1 to 1:7. The THD+N of the converted signal is around −120 dB. The core uses around 700 Slices when implemented on a *Xilinx Kintex-7* Field Programmable Gate Array (FPGA) board as a stereo converter. This IP has some limitations that can be improved: Firstly, the configurable nature of the IP should allow it to support more channels at the cost of more resources if needed. The THD+N of the output can be further reduced to make the core more competitive with its Integrated Circuit (IC) counterparts. The same reason justifies the improvement of the limit imposed on the ratios. Finally, as the core replicates itself for each pair of channels, the resource usage per channel can be further reduced.

The overall growth of the market for embedded audio systems, with many requiring sample rate conversion, motivates the development of a better sample rate converter IP core. The only existing IP implementation of an ASRC, the CWda52, shows several limitations, so it makes sense to develop an improved IP core, competing with both the existing IP core and the IC counterparts.

The main objective of this work is to design an asynchronous sample rate converter (ASRC) IP core using the Verilog hardware description language. The core should meet the following specifications: (1) Support for up to 24-bit samples; (2) conversion from and to any sample rate, in the range between 8 kHz and 192 kHz; (3) capability of converting multiple channels, limited by the operation clock frequencies; (4) output THD+N equal or lower than -130 dB (for 24-bit samples); (5) synchronization time equal or lower than 200 ms; (6) variation of phase between input and output after a reset of less than one output sample; (7) resource consumption per channel of the same order as the CWda52, or lower.

2. Background

2.1. Theoretical Operation

The ideal ASRC converts a discrete time input signal $x[n]$, sampled at a rate F_{s1}, to a continuous time signal $x(t)$, with the use of a reconstruction filter. The signal is then filtered by an anti-aliasing filter which ensures that its output $y(t)$ has no components which would violate Nyquist's law. Signal $y(t)$ is then converted to a discrete time signal $y[m]$, sampled at the desired output rate F_{s2} [8]. One of the challenges of creating a purely digital solution is the design of the LPF digital filter, which is at one time accurate and efficient. The classical approach to the problem consists in upsampling the signal by a factor L (interpolation), doing the processing at the frequency $L \times F_{s1}$, and downsampling the result by a factor M (decimation), as a means to emulate a discrete to continuous and continuous to discrete signal conversion [13]. A simple block design of the algorithm is shown in Fig. 1.

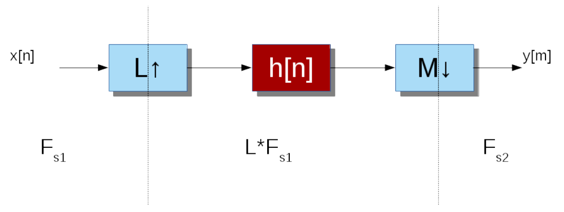


Figure 1: Classic digital sample rate conversion algorithm by factor L/M

In practice, an input signal $x[n]$, sampled at frequency F_{s1} is upsampled by insertion of $L-1$ samples with value zero between two consecutive input samples. The resultant signal is then inserted into a low pass filter $h[n]$, which will interpolate the inserted samples and avoid aliasing. Finally, the output of the filter is downsampled by taking only every M-th sample for the output signal $y[m]$. The relationship between L and M is such that

$$F_{s2} = \frac{L}{M} F_{s1}. \tag{1}$$

Regarding the filter $h[n]$, its cutoff frequency depends on the relationship described in Equation (1). In the upsampling case ($L \geq M$), there is the need to remove the resultant spectral images, using a filter with a normalized cutoff frequency ($\Omega_c \leq 0.5\pi/L$). In the case of downsampling ($L < M$), there is the need to filter signal frequencies that would cause aliasing, leading to a filter with a normalized cutoff frequency ($\Omega_c \leq 0.5\pi/M$). By joining the two conditions, the resultant filter should have a cutoff frequency

$$\Omega_c = min(\frac{\pi}{2L}, \frac{\pi}{2M})[rad]. \tag{2}$$

In the current state of the art, this algorithm is the basis of most synchronous and asynchronous sample rate converters. For conversions which use small values of L and M the computational effort is modest. However, if the conversion involves sampling rates with a small difference, the factors L and M will increase significantly, increasing drastically the computational cost of the conversion, only to have most of the computed samples discarded. Note that in this case a small normalized cutoff frequency must be used for the filter. Furthermore, one needs to design the filter to both remove aliasing and interpolate the $L-1$ inserted samples. This is why design of the filter $h[n]$ is the main challenge of this architecture. The solution presented in this thesis is based on the use of a fractional delay filter. Additionally, for asynchronous sample rate converters, the filter is not only time-varying, but also varies with the sample rate ratio. This means that there is the need to define a structure that computes the ratio and adapts the filter. For synchronous sample rate converters, the ratio stays constant. This can lead to a predictable filter, leading to the possibility to trade off storage space for computation time, by precomputing a finite set of filters [4]. Since both the reconstruction and anti-aliasing filters are linear systems, they can be combined together in a single filter whose frequency response $H_d(e^{j\Omega})$ is the product of the frequency responses of the two filters:

$$H_d(e^{j\Omega}) = \begin{cases} 1, |\Omega| < \Omega_c \\ 0, otherwise \end{cases}. \tag{3}$$

To obtain the impulse response of filter $h[n]$, one performs the inverse discrete-time Fourier transform (IDTFT) of $H_d(e^{j\omega})$.

$$h[n] = \frac{\Omega_c}{\pi} sinc\left[\frac{\Omega_c}{\pi}(n - \tau_d)\right]. \tag{4}$$

As this function is infinite and non-causal, an approximation must be done while retaining enough low-pass filtering capability to ensure quality. Some methods to approximate of the filter include windowing the ideal filter, applying Lagrangian interpolation to compute an intermediate coefficient from a set of known filter samples, etc. These and other methods are explained in detail in [12, 17, 18, 19]. The resultant filter is a section of the sinc function which contains a certain number of zeroes.

2.2. State Of The Art

Multiple distinct implementations of the filter can be performed, leading to different results regarding output fidelity and resources usage. The most direct approach is to perform the direct computation of the discrete convolution between the filter and the samples, using two memories for the coefficients and samples, and a MACC unit for the computations [1]. A block diagram of the MACC unit is shown in Fig. 2.

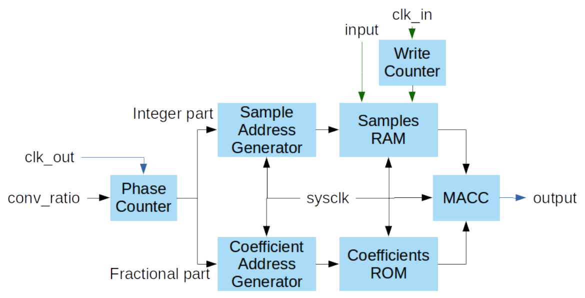


Figure 2: MACC filter block diagram.

In this implementation, an accumulator generates the phase of the desired output samples, by accumulating the sample rate conversion ratio. The integer part of the accumulated value will correspond to the previous input sample that is closest in phase to the current output sample. The fractional part is the distance between the phase of the desired output sample and the closest input sample, and is used to select the starting point of the filter. Graphically, this selection is represented as centering the sinc to the closest input sample. It can also be interpreted as a polyphase filter [8]. After the starting values are determined, both the coefficients and samples addresses are decremented in equivalent values, obtaining the previous samples and respective coefficients. The output sample $y[n]$, is obtained by

$$y[n] = \sum_{i=0}^{N} a_i x[n-i], \tag{5}$$

where a_i are the coefficients of the filter and $x[n-i]$ is the input sample at phase $n-i$. This design has the advantage of being the easiest to implement, as well as being the one that uses the least amount of logic in comparison to the other implementations presented in this chapter, requiring only one MACC unit for the computations. However, it requires a large amount of filter coefficients, which, in hardware, results in high memory usage, an undesirable result. The problem can be solved by interpolating the filter's coefficients, leading to the use of some logic to reduce the amount of memory used, as some intermediate values do not need to be stored in it. The MACC unit is the implementation used for the work performed. Other possible implementations include a Cascaded Integrator Comb filter [6], a approximation of the piece-wise sinc function into quadratic functions [14, 2], or a Farrow Structure [9].

The implementation of the filter allows one to design a synchronous sample rate converter. To implement its asynchronous counterpart, the sample rate conversion ratio needs to be computed. The sample rate conversion ratio can be expressed by

$$\rho = \frac{F_{sOut}}{F_{sIn}}, \tag{6}$$

where ρ is the sample rate conversion ratio, and F_{sIn} and F_{sOut} are the respective sample rates of the input and output signals. There are two possible implementations for the estimator of ρ. The first implementation uses multiple accumulators to compute the input and output sample periods. This implementation has no error tracking mechanism, leading to a drift of the group delay of the converter, due small errors in the computed value of ρ. An additional module to avoid the drift needs to be implemented. The second implementation is the use of a digital phase locked loop (DPLL) as a frequency tracker [15, 16]. Similarly to a conventional PLL, the DPLL can be split into three blocks: a phase detector, a loop filter and a voltage controlled oscillator (VCO). The loop filter is a low-pass filter, and it attenuates the effect of the phase detector's jitter, at the cost of tracking speed.

3. Implementation

Fig. 3 illustrates the symbol and interface signals of the sample rate converter designed. The green signals are defined in the *audio_in_mclk* domain, while the blue signals are defined in the *audio_out_mclk* domain. The remaining signals are in the system clock domain.

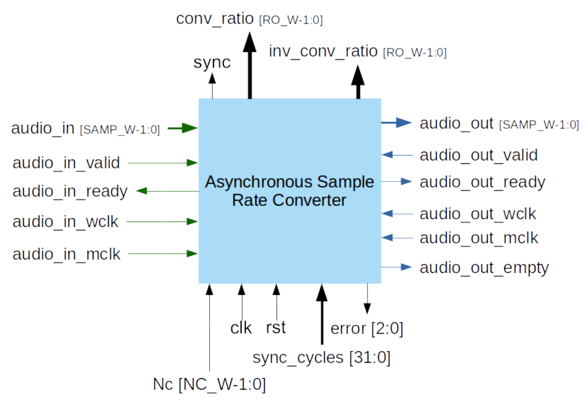


Figure 3: Symbol and interface diagram.

Table 1 presents the synthesis parameters available to configure the synthesis of the core.

Parameter	Default Value
SAMP_W	24
NC_W	8
RO_W	35
SAMP_BUF_W	10

Table 1: Synthesis Parameters

The ASRC is divided in three main modules: the Input Data Memory, the Ratio Estimator and the Resampler. Additionally, the ASRC includes a positive edge detector circuit, implemented with a register, to generate the *start* signal, used to start the computation of a new output sample, as well as an asynchronous FIFO, used to temporarily store the output samples to be read, and to allow their crossing from the system clock to the output audio master clock. Fig. 4 illustrates the block diagram of the core.

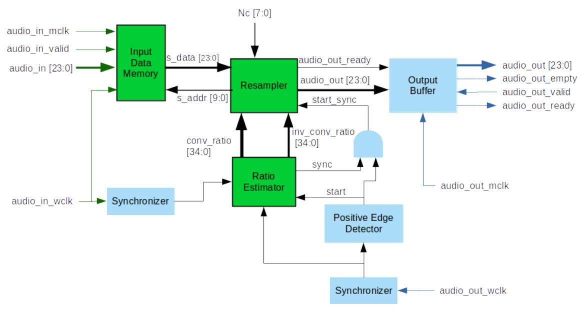


Figure 4: Asynchronous sample rate converter top block diagram.

3.1. Input Data Memory

The data memory is a circular dual port Random Access Memory (RAM) capable of working at two different clock domains. In the input clock domain, the samples *audio_in* are stored. A counter controls the address at which the samples should be written. In the system clock domain the samples are read by accessing the position given by *s_addr*. The read samples are used to perform the filtering. For Nc Time Division Multiplexed (TDM) channels, the address of the next sample of a certain channel, $s\_addr[n+1]$, is given by

$$s\_addr[n+1] = s\_addr[n] + Nc. \tag{7}$$

3.2. Conversion Ratio Estimator

In this work, the use both techniques presented in section 2.2 constitutes the conversion ratio estimator. The period measurement and averaging perform the first approximation, obtaining the approximate conversion ratio, $\hat{\rho}$, and the approximate inverse of the conversion ratio $\hat{\rho^{-1}}$.

As the error in the approximations leads to a deviation of the read and write pointers of the data memory, as well as a variable group delay of the filter, a DPLL, used as a frequency tracker, performs a finer adjustment of the approximation of $\hat{\rho^{-1}}$, by using the variation of the average delay measured by the phase detector, $\Delta D = 2\pi(\phi_{in} - \phi_{out})$, to apply corrections. The variation of the delay between the input and output phases, ϕ_{in} and ϕ_{out} respectively, over a certain time period, Δt is given by

$$\frac{\phi_{in} - \phi_{out}}{2\pi F_{sIn}} = \Delta t - \rho \times \hat{\rho^{-1}} \times \Delta t. \tag{8}$$

By adapting to discrete time, one can obtain the value of the correction, C, needed to apply, by ensuring that ΔD converges to zero,

$$C = \frac{\Delta D}{N_{out}}. \tag{9}$$

To allow a smoother variation of $\hat{\rho^{-1}}$, the value of the correction is attenuated.

The conversion ratio estimator is controlled by a finite state machine (FSM). Initially, the FSM enables the period meters and allows them to accumulate for a certain time, performing the average of the measured periods, allowing the computation of $\hat{\rho}$ and $\hat{\rho^{-1}}$. Afterwards, the FSM enters a loop that performs the measurement and average of the delay, using it to periodically apply new corrections.

3.3. Resampler

The resampler is the main module of the sample rate converter, and is implemented with a MACC unit, as explained in Section 2.2. The resampler splits into three submodules: address generator, coefficient memory and multiply-accumulator.

The address generator computes the addresses of both the input samples used to compute the output sample, and the addresses of the corresponding coefficients. A filter setup block computes the address of the current output sample, the initial address of the coefficient to be used, α (this address changes depending on the side of the sinc that is being currently used), and the parameter h_step, a parameter related to the normalized cutoff frequency of the filter. The integer part of the address of the current output sample is the base address of the input samples to be used. A accumulator that accumulates on increments of the number of channels, Nc, outputs the address of the offset to be applied to the base. Depending on the side of the sinc the offset can be positive or negative. The address of the coefficient comes from an accumulator that is initialized with α and accumulates in increments of h_step. When the accumulated value reaches the end of the coefficient memory, the side of the sinc function switches, and the coefficient accumulator resets with the new value of α.

The coefficient memory is a read-only memory (ROM) that contains the truncated *sinc* function. To reduce the amount of coefficients stored, the module also contains a linear interpolator. When a coefficient $h[i+\Delta]$ is needed, $h[i]$ and $h[i+1]$ exist in the lookup table, and Δ is a fractional positive distance from i, the coefficient is obtained by

$$h[i+\Delta] = h[i] + \Delta(h[i+1] - h[i]). \tag{10}$$

As its name indicates, the multiply-accumulator implements the multiply-accumulate function. The accumulated values are the products between the coefficient and the correspondent input sample. The initial value of the accumulator is set by directly loading (not accumulating) the first product. The final register stores the output sample *audio_out*, and is only enabled when all accumulations have finished. The output sample is multiplied by h_setp, as the filter is normalized for unit gain in the pass-band.

To optimize the implementation of the ASRC, allowing it to run with a system clock frequency of at least 100 MHz on a low cost FPGA, 9 pipeline registers are added: 3 in the address generator module, 4 in the coefficients ROM module, and 2 in a MACC module.

4. Test Environment

4.1. Signal Generation and Analysis

To test the hardware developed, there is the need to generate the input samples, as well as the filter's coefficients. Furthermore, with the knowledge of the input signal used, the output should be analyzed. For this matter, four *Octave* scripts are used: one to generate the filter's coefficients, one to generate the input signal, one to analyze the output signal and one to test the effect of resetting the core on the group delay.

The script that generates the input samples receives the input and output sampling rates, and the frequency and magnitude of the test signal to be generated. The script also adapts the value of the sampling rate to the closest values that can be obtained by the FPGA's clock generator, using master clocks with periods of 42 ns and 91 ns. It also adjusts the frequency of the test signal, allowing it to be a divisor of the output sampling rate. To guarantee that the output has enough samples for analysis, the input signal generated has enough samples to ensure a minimum of 120 signal periods and 11000 output samples. The test wave generated is a sine wave.

To generate the coefficients, the script receives three values: $filter\_nzeros$, $filter\_nfrac$, and H_bits, which give the number of zeroes of the sinc function, the number of address bits to address the filter's memory, and the number of bits to quantify the value of the coefficients, respectively. The filter generates an ideal sinc function and truncates it using a *Kaiser* window. The values of $filter\_nzeroes$, $filter\_nfrac$ and H_bits are 32, 10 and 24, respectively.

The analyze the output, the script computes its spectrum, through a Fast Fourier Transform (FFT). Afterwards, it computes the THD+N by summing the power value of every bin which is not deemed to belong to the original sinusoidal signal. It also allows a small tolerance around the signal bin's frequency to compensate for the difficulty to avoid scattering when computing spectra.

The script that analyzes the effect of resetting the core on the group delay receives two versions of the output: one that ran without a reset during the conversion, and one that ran with a reset during the middle of the conversion. The script compares the results and computes the approximate phase between them, expressed as

$$\phi = \arccos \frac{\overrightarrow{y_1} \cdot \overrightarrow{y_2}}{|y_1||y_2|}, \tag{11}$$

where y_1 and y_2 are vectors containing each sample as an element. For a number of samples large enough, the approximation is accurate, as explained in [10].

4.2. IOb-SoC Hardware Platform

IOB-SoC [11], an open-source system-on-a-chip (SoC) platform, tests the implementation of the ASRC in FPGA,. The block diagram of the SoC is shown in Fig. 5.

The PicoRV32 [?], a RISC-V soft processor, controls the SoC. The CPU can access an internal static

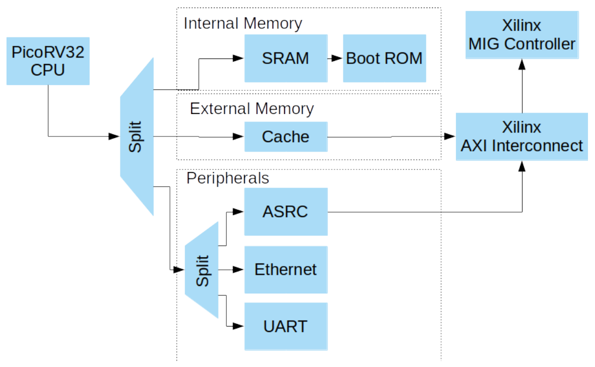
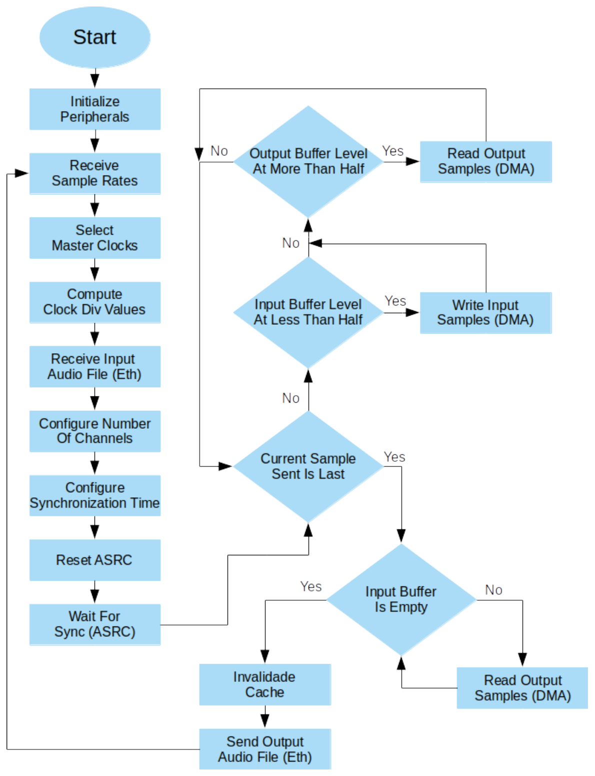


Figure 5: IOB-SoC block diagram.

RAM, an external Double Data Rate (DDR) memory (through a cache), and three peripherals:

- the ASRC, connected through a wrapper that uses registers, FIFOs and a Direct Memory Access (DMA) module to configure the core, as well as send and receive the samples.
- the UART, used to send runtime messages to the user.
- the Ethernet module, used to receive the input samples from the PC, and send the output samples.

Fig. 6 illustrates the flowchart of the firmware ran by the CPU.



Start
Initialize Peripherals
Receive Sample Rates
Select Master Clocks
Compute Clock Div Values
Receive Input Audio File (Eth)
Configure Number Of Channels
Configure Synchronization Time
Reset ASRC
Wait For Sync (ASRC)
Output Buffer Level At More Than Half
Yes
No
Read Output Samples (DMA)
Input Buffer Level At Less Than Half
Yes
No
Write Input Samples (DMA)
Current Sample Sent Is Last
Yes
No
Input Buffer Is Empty
Yes
No
Invalidade Cache
Read Output Samples (DMA)
Send Output Audio File (Eth)

Figure 6: ASRC testing firmware flowchart.

5. Results

To validate the Asynchronous Sample Rate Converter design, a rigorous testing suite has been applied to it, based on similar tests that can be found in [3], [5] and [7]. All tests have been applied to an FPGA implementation of the design, embedded in the SoC described in Section 4.2, and run on a Kintex Ultrascale FPGA (XCKU040-FBVA676-1-C) device.

The Fast Fourier Transform (FFT) method is the main process to obtain the results, as most of the rely on the magnitude of the signal, or the sum of magnitude of the noise. By default, the input signal is a sine wave with a magnitude of −1 dB, and a frequency of 1 kHz. Most of the performed tests ran 16 times, using combinations of 44.1 kHz, 48 kHz, 96 kHz, 192 kHz for the input and output sample rates. The tests for the Total Harmonic Distortion Plus Noise (THD+N) and group delay ran for every combination of the 11 most commonly used sample rates in audio applications, from 8 kHz to 192 kHz. All of the tests met the specification defined in Section 1.

The plot of the FFT of the output signal for a conversion from 44.1 kHz to 48 kHz is shown in Fig. 7. Note that the magnitude of the signal bin is lower than the expected value. This is due to the window applied to the FFT, used to eliminate spectral leakage.

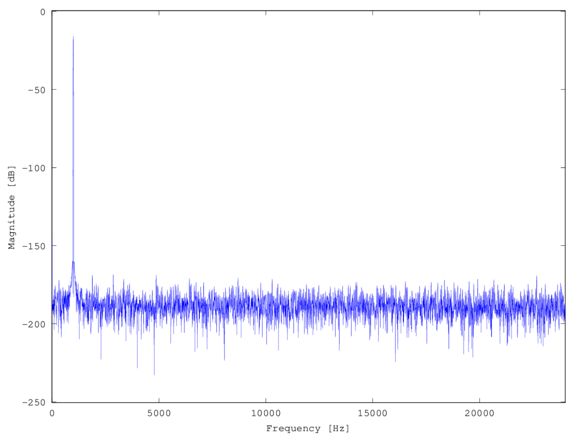


Figure 7: FFT of a conversion from 44.1 kHz to 48 kHz.

The THD+N obtained for a subset of the conversions tested is shown in Table 2.

The results shown in 2 show that that the specification of a THD+n equal to −130 dB or less is fulfilled for some the most common sampling rate conversions.

Fig. 8 shows the THD+n results when varying the input signal's frequency, for a single conversion.

Fig. 8 shows that the THD+n increases for signals with frequencies close to the Nyquist frequency

Fin [Hz]	Fout [Hz]	THD+N [dB]
8000	177242	-139.706173
11022	96006	-138.768504
44132	48003	-138.784562
177242	192012	-141.533573
192012	11022	-141.887817
87912	8000	-141.829588
48003	32002	-136.135586
11022	8000	-138.289522

Table 2: Total harmonic distortion+noise ratio for some conversions

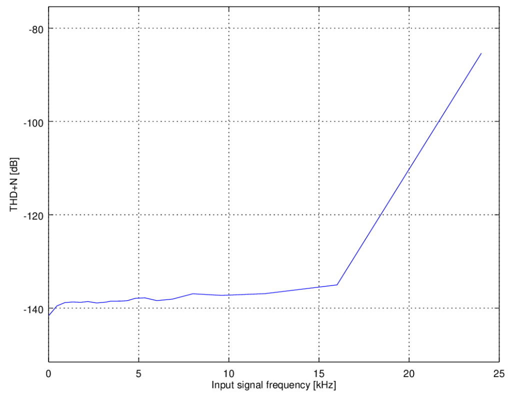


Figure 8: THD+N with varying input signal frequency for conversion from 44.1 kHz to 48 kHz.

($F \approx F_{in}/2$). This is caused by aliasing in the filtered output, as the low-pass filter is not selective enough to completely remove the distortion caused by aliasing.

The THD+n for a subset of conversions of a signal with a frequency of 1 kHz and varying magnitude, between -120 dB and -1 dB, is shown in Fig. 10.

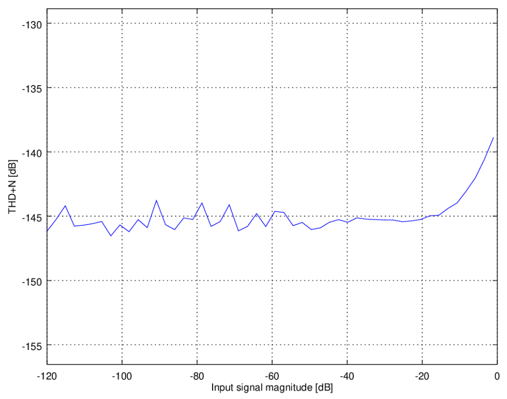


Figure 9: From 44.1 kHz to 48 kHz.

Figure 10: THD+N of the ASRC output for fixed conversions while varying the input magnitude.

The increase of the input magnitude leads to an increased THD+N. This is due to the increase of the harmonics' amplitude, as well as the spread of some of the energy of the signal to adjacent frequencies, caused by the adjustments of the conversion ratio, performed by the core. In these conditions, the specification is still met.

Fig. 11 shows the frequency responses of a conversion from 44.1 kHz to 48 kHz.

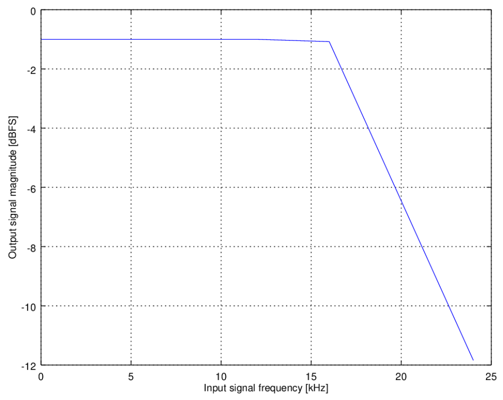


Figure 11: Frequency response of the ASRC for a conversion from 44.1 kHz to 48 kHz.

The results of the phase difference after reset are obtained using the method explained are shown in

Table 3.

Fin [Hz]	**Fout [Hz]**	**Phase Shift (# Output Samples)**
11022	96006	0.329150
44132	48003	0.001540
176366	192012	0.090451
192012	11022	0.009258
88183	8000	0.044237
48003	32002	0.333216
11022	8000	0.044391

Table 3: Group delay of some conversions.

Although the synchronization process leads to a small change in the group delay, due to the lack of a phase tracker, displacement is less than one output sample. As such, it presents no audible effect.

The ASRC is considered linear, as the amplitude of the output signal is equal to the amplitude of the input signal. Table 4 shows the value of the proportionality constant, β, and coefficient of determination R^2, when performing a linear regression on some conversions.

Fin [Hz]	**Fout [Hz]**	β	R^2
192012	44132	0.999999	99.99996%
192012	96006	1.000026	99.99997%
44132	48003	1.000071	99.99993%
44132	96006	1.000057	99.99994%
48003	192012	0.999945	99.99990%
48003	44132	0.999785	99.99993%
96006	192012	0.999981	99.99992%
96006	48003	0.999930	99.99995%

Table 4: Results of linear regression of some conversions

Table 4 allows one to conclude that the filter is linear for the range of input magnitude tested, as the value of R^2 is close to 100%. The value of β shows that the gain of the ASRC is close to 1.

The detailed resource usage of the ASRC on the Kintex Ultrascale FPGA (XCKU040-FBVA676-1-C) is shown in Table 5.

Resource	**Used**
LUTs	1654
Registers	1540
DSPs	9
BRAM	13

Table 5: Resource utilization on a XCKU040

More than half of the lookup tables (LUTs) are used by the ratio estimator (ro_meter) module. In it, the divider module is the most resource-hungry, as it needs to compute averages that require many bits to account for the accumulated periods, number of accumulations and of fractional bits of the conversion ratio.

The core also uses 9 digital signal processing blocks (DSPs) for its multipliers. In an ASIC implementation, the multipliers would occupy a significant silicon area, most likely higher than the silicon area taken by the ratio estimator module.

Regarding the usage of memory blocks, the coefficient ROM is the element that uses the most, as the module stores 16384 (2^{14}) samples, with 24 bits per sample, stored. In FPGA, ROMs are typically implemented with pre-initialised block RAMs, but in an ASIC implementation, the implementation of ROMs leads to an area one order of magnitude lower than the implementation of RAMs.

There are two components that lead to the limitation of the number of channels: the size of the data memory, presented in Subsection 3.1, and the number of clock cycles available to compute a sample for every channel.

Considering that the filter is a *sinc* function with 32 zeroes and the coefficients are iterated in increments of *h_step*, and acknowledging that one needs one input sample for each coefficient, size of the data memory, which should have a capacity to store enough input samples for the computation of an output sample per channel, should be given by the expression

$$Mem\_Size \geq \frac{32}{min(0.875, \rho)} \times N_{channels}. \qquad (12)$$

That size can, however, be changed through the synthesis of the core with a different synthesis parameter.

As the MACC module performs a single multiplication per system clock cycle,the output samples can only be computed if the system clock cycle is $(N_{coeffs}+10) \times N_{channels}$ times faster than the output sample rate. The 10 extra clock cycles refer to the 9 cycles needed to allow the internal signals to pass through the pipeline registers, as well as an extra cycle needed to change the side of the filter.

6. Conclusions

An ASRC is a rather complex circuit, and only major semiconductor players such as Cyrrus Logic, Analog Devices and Texas Instruments have Integrated Circuit (IC) solutions available in the market. To the best of the author's knowledge, the only existing ASRC IPs in the market, the CWda5x family, is made available by the IP design company Coreworks, SA. As was explained in Section 1, the CWda5x cores have a higher (worse) THD+N than the IC solutions, as well as a more limited range of conversion ratios.

This thesis proposes a new design of an audio Asynchronous Sample Rate Converter (ASRC), implements it and presents the experimental results. The circuit is described in Verilog, simulated, and prototyped in FPGA. The ASRC is designed as a multi-channel sample rate converter Intellectual Property (IP) module for integration into System-on-Chip. The imposed specifications have the purpose of making it competitive to any IC or IP solutions, with the CWda52 IP core, the AD1896 and the SRC4194 chips being the leading competitors.

The specifications include: support for up to 24-bit samples, sampled in the range between 8 kHz and 192 kHz, multiple (hundreds) of channels, THD+N of −130 dB or less, a synchronisation time lower than 200 ms, variation of phase between input and output of less than one output sample after a reset, and a hardware resource consumption similar to the CWda52.

The bit-width of the samples can be specified as a synthesis parameter. Although the specification imposes the upper bound of 24 bits, the IP can support any bit-width, though tests for more than 24 bits are not performed.

Similarly to the CWda5x IP core family and the IC chips, the present core supports any sample rate between 8 kHz and 192 kHz. The present design has no limit regarding the conversion ratio; the supported sample rates define the conversion ratio limits. Hence, the supported range is from 24:1 to 1:24 ratios. This range is broader than the ranges of any other solution, as the maximum previously found by the author is 16:1 to 1:16 [5].

The core supports multiple channels. The maximum number of channels supported depends on the input and output sample rates, data memory size, and system clock's frequency. While one can change the size of the data memory by changing the synthesis parameters, the frequency of the system clock may be limited by its critical path length. For example, the core can support up to 8 channels for the worst conversion ratio: 192:8 kHz for a 100 MHz system clock For most standard conversion rates, the core support at least 16 channels. The solution is thus the most competitive ones in terms of the number of supported channels.

The output's THD+Nis lower than −136 dB for all the conversions in the supported range; a total of 121 conversions have been run. For most conversions, the THD+N is around −140 dB. For the best case, the THD+N is −143 dB. These figures are a direct improvement over the CWda52 and lead to a sound fidelity similar to the IC counterparts, as their THD+N reaches values around −140 dB [5].

Thanks to the unique architecture of the conversion ratio estimator, the synchronisation time of the core is only 20 ms. This low synchronisation time improves the IC solutions, as the synchronisation time is one order of magnitude lower. This architecture also presents no issues regarding the variation of the phase after a reset; it is lower than one output sample for the 121 tested conversions.

For two channels, the FPGA implementation of core uses around 60% of the total amount of LUTs used by the CWda52, while the DSP usage is double. However, for more than two channels, the resource usage of the CWda52 increases as it replicates itself for each pair of channels. The core proposed in this thesis has no hardware overhead with the number of channels, which makes the hardware resource usage per channel of the proposed core lower and lower than the CWda52's as the number of channels increases.

By changing the synthesis parameters, ASRCs with different specifications can be produced, trading off its high-end features for less hardware resource usage.

Once the conversion ratio is estimated, the computation of the output samples may proceed synchronously. Synchronous sample rate converters find many applications in digital audio processing to operate on stored signals for which the sample rate is known. Consequently, synchronous sample rate converter IPs can easily be implemented using just the resampler part of this work and adding the necessary system integration circuitry.

The asynchronous nature of the algorithm dictates the need to work with multiple clock domains, which is as complex as it can get in terms of *digital* hardware design. In this work, three different clock domains are used: the input, output and processing clock domains. Typical synchronisation structures have been developed as needed to avoid errors caused by metastability or data misses.

References

[1] R. Adams and T. Kwan. A stereo asynchronous digital sample-rate converter for digital audio. *IEEE Journal of Solid-State Circuits*, 29(4):481–488, 1994.

[2] N. Aikawa and Y. Mori. Kernel with block structure for sampling rate converter. In *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03).*, volume 6, pages VI–269, April 2003.

[3] Analog Devices. *192 kHz Stereo Asynchronous Sample Rate Converter*, March 2003. Rev. A.

[4] P. Beckman and T. Stilson. An efficient asynchronous sampling-rate conversion algorithm for multi-channel audio applications. In *AES Convention Papers Forum*, number 6553, October 2005.

[5] Burr-Brown Products from Texas Instruments. *4-Channel, Asynchronous Sample Rate Converter*, June 2004. Rev. B.

[6] S. CHARANJIT, M. Patterh, and S. Sharma. Efficient implementation of sample rate converter. *International Journal of Advanced Computer Sciences and Applications*, 1, 01 2011.

[7] coreworks. *Multi-Channel Audio Sample Rate Converters*, June 2016.

[8] R. Crochiere and L. Rabiner. *Multirate Digital Signal Processing.* Prentice-Hall Signal Processing Series: Advanced monographs. Prentice-Hall, 1983.

[9] C. W. Farrow. A continuously variable digital delay element. In *1988., IEEE International Symposium on Circuits and Systems*, pages 2641–2645 vol.3, June 1988.

[10] Fat32. Calculating the phase shift between two signals based on samples. `https://dsp.stackexchange.com/questions/41291`, 2017.

[11] IObundle Lda. IOb-SoC. `https://github.com/IObundle/iob-soc`, 2020.

[12] P. J. Kootsookos and R. C. Williamson. Fir approximation of fractional sample delay systems. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 43(3):269–271, March 1996.

[13] S. K. Mitra and J. F. Kaiser, editors. *Handbook for Digital Signal Processing.* John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1993.

[14] Y. Mori and N. Aikawa. Kernel using piecewise nth polynomials for rate converter. In *Proceedings of the 6th Nordic Signal Processing Symposium, 2004. NORSIG 2004.*, pages 57–60, June 2004.

[15] F. Rothacher. Sample rate conversion: algorithms and vlsi implementation. 1995.

[16] E. Stikvoort. *Some subjects in digital audio : noise shaping, sample-rate conversion, dynamic range compression and testing.* PhD thesis, Department of Electrical Engineering, 1992. Proefschrift.

[17] C. Tseng and S. Lee. Design of fir fractional delay filter based on maximum signal-to-noise ratio criterion. In *2013 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*, pages 1–8, Oct 2013.

[18] V. Valimaki and T. I. Laakso. Principles of fractional delay filters. In *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.00CH37100)*, volume 6, pages 3870–3873 vol.6, June 2000.

[19] A. Yardin, G. D. Cain, and A. Lavergne. Performance of fractional-delay filters using optimal offset windows. In *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 3, pages 2233–2236 vol.3, April 1997.