Design of a Digital CIC Filter for an Audio Sigma-Delta ADC in the Scilab Environment

Otávio Elias V. de Freitas, Edivania F. Silva, Crístian Müller and Paulo César C. de Aguirre.
Computer Architecture and Microelectronics Group - GAMA
Federal University of Pampa - UNIPAMPA, Alegrete, Brazil
{otaviofreitas.aluno, edivaniaferreira.aluno, cristianmuller, pauloaguirre}@unipampa.edu.br

***Abstract*—Sigma-Delta ADCs are widely adopted in high-resolution applications. They are composed of an analog loop filter and a digital decimation filter. The decimation filter operation is usually simulated using the Matlab/Simulink environment. However, the cost of commercial behavioral simulators is high, not being accessible to everyone. Thus, this paper presents the design of a Cascaded Integrator-Comb (CIC) filter for a 3rd-order Continuous-Time Sigma-Delta Modulator using an open-source tool called Scilab. The CIC filter was initially designed in the Simulink environment using ideal blocks. Then, the same CIC filter was designed in the Scilab environment. The Scilab modeling matches the Simulink implementation, and the detail implementations are explored in this paper.**

I. Introduction

Analog-to-digital converters (ADCs) play an important role on signal processing since they provide a bridge between analog and digital worlds [1]. The Sigma-Delta ADC is one of the most used ADC architectures for high-resolution applications such as industrial instrumentation [2] or audio acquisition [3]. Smartphones and current wearable devices such as smartwatches require audio acquisition systems. In the last decade the use of microelectromechanical systems (MEMs) microphones have became a leading solution to the audio module in most portable devices [4]. Sigma-Delta ADCs are adopted in most MEMs microphone systems and the current state-of-the-art MEMS microphones show an SNR ranging from 67 to 69dB [5].

Sigma-Delta ADCs are composed of an analog loop filter and a digital decimation filter [6]. The loop-filter sets the ADC main specifications as the signal-to-noise distortion ratio (SNDR) and the effective number of bits (ENOB). The sigma-delta modulation is based on oversampling and noise-shaping. The noise-shaping effect is responsible for increasing the noise power at high-frequencies, outside the signal bandwidth. To remove this noise, a decimation filter must be employed. This filter is usually composed of a Cascaded Integrator-Comb (CIC) filter followed by one or more Finite Impulse Response (FIR) filters [7]. The purpose of the FIR filters are to compensate for the CIC filter attenuation near the cutoff frequency and to increase the off-band noise removal, providing no signal distortion. Finally, the goal of using more than one FIR filters is to improve the efficiency of the design in terms of area and power consumption.

The implementation of these filters through simulation is very important, so Matlab stands out as a powerful calculation tool that covers several areas. In order to find a cheap and efficient alternative, open source tools arise, which aim to help small companies trying to enter the market and are faced with high values for software. The use of open source software is not only linked to small companies, large companies such as Hutchinson and Peugeot support and invest in Scilab, which one exists since 1980 and has the support of more than 10 large companies.

The goal of this paper is to show that we can implement digital filters, (in this case the CIC filter previously implemented in Systemverilog and Matlab/Simulink as discussed in the paper [8]), with open software tools such as Scilab/Xcos.

This paper is organized as follows: Section I presented a brief introduction. Section II explains the continuous time sigma-delta modulator used as a case study. The CIC filter is introduced in Section III. The implementation in Scilab is covered in Section IV while simulation results of the CIC filter are presented in Section V. Finally, some conclusions are drawn in Section VI.

II. Sigma Delta Modulator

This work considers the same study case as the one in the previously published paper [8]: A third-order single-bit continuous time sigma-delta modulator. This modulator is presented in Figure 1.

The complete modulator information is also provided in this paper as follows: the coefficients are [c_1, c_2, c_3, c_a, a_2, a_3, g_1] = [1, 1, 1, 0.668, 0.244, 0.044, 0.000592]. The clock signal of this modulator is 4 MHz and the signal bandwidth is 20 kHz. The modulator was simulated for an 1.4038-kHz input signal with amplitude of -6.02 dBFS and -40 dBFS. Its output power spectrum density is shown in Figure 2. For the -6.02-dBFS input signal the signal-to-noise-and-distortion ratio (SNDR) is

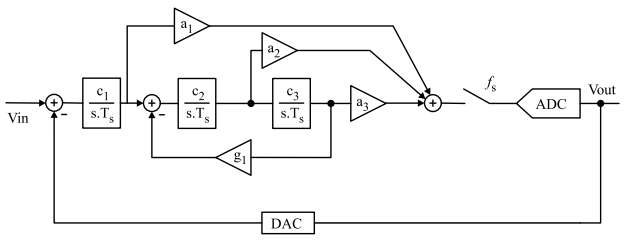

Fig. 1. Third-Order Continuous-Time Sigma-Delta Modulator.

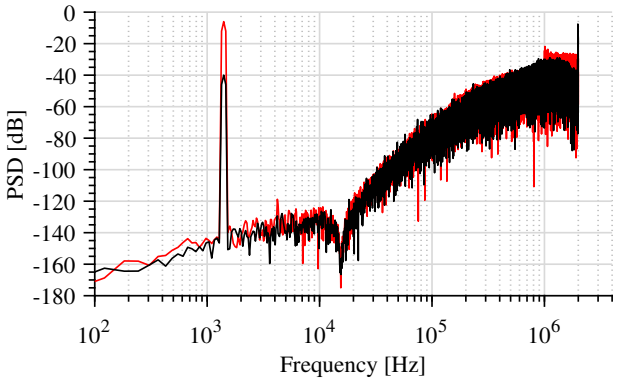

Fig. 2. Power spectrum density of the modulator output for a 1.4038-kHz input signal with amplitude of -6.02 dBFS (black) and -40 dBFS (red). 65,536-points FFT.

101.16 dB and the effective number of bits (ENOB) is 16.51 bits. For the -40 dBFS input signal the modulator presents and SNDR of 70.27 dB and an ENOB of 11.35 bits. The sigma-delta modulator defines the sigma-delta ADC ENOB, and the decimation filter has a number of bits higher than the modulator ENOB.

The bitstream at the output of this modulator is saved in a text file and used as input stimuli in the CIC filter testbench in both Simulink and Scilab for comparison purposes.

III. CIC Filter

This work presents the development of a CIC filter similar to [7] and already explored by the authors in [8]. This filter has order four and its transfer function is shown in eq. 1, where N is the decimation factor and M is the filter order.

$$H(z) = \left(\frac{1 - z^{-N}}{1 - z^{-1}}\right)^{M} \tag{1}$$

It is composed of four integrators and four differentiators, these are separated by a clock delay of 16. The clock reduction determines the filter decimation rate. The purpose of the CIC filter is to reduce the reporting rate of the signal to the desired signal rate. The filter general block diagram is shown in Figure 3, and its frequency response is shown in Figure 4.

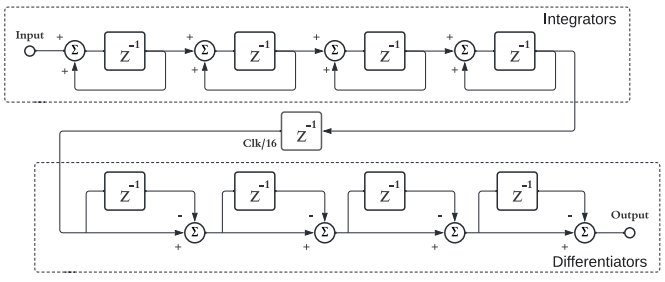

Fig. 3. Block diagram of the 4-th order CIC filter.

Initially this filter was designed in Matlab/Simulink as shown in Figure 5. The output of the modulator shown in Figure 1 was used with stimulus at the entrance of the designed

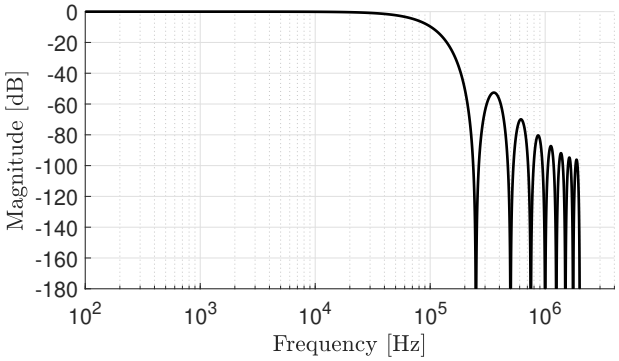

Fig. 4. CIC filter frequency response.

filter. The CIC filter is tested for considering the modulator output for the input signal amplitude of -40 dBFS. After that, these modulator output data were introduced as a stimulus for the CIC filter implemented in Scilab.

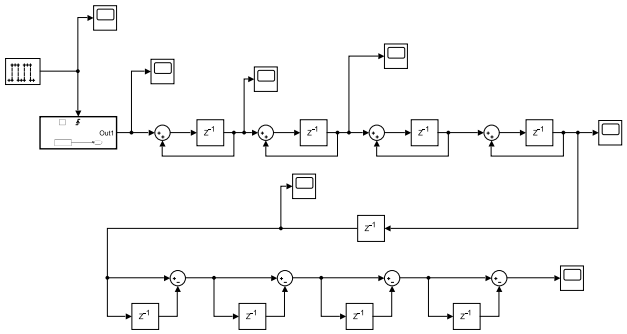

Fig. 5. CIC filter implemented in the Matlab/Simulink environment.

IV. CIC filter in Scilab

A. Integrator

With Simulink we use the block Z^{-1} and an adder to design the integrator, then connecting them as shown in the Figure 5. In Scilab, the delay block receives the name $1/Z$ and we need to convert the variable types to match each block specification, since some blocks only work with variables of type double, and our operations must be done with type int16. By contrast, in Simulink this conversions was automatic.

We can see in the Figure 6 the integrator implemented in Scilab, it has one event input (in red) that controls the majority of the time variables, one data input (in black) and one data output. This circuit is inside a super block, that is a great advantage of the software, allowing for an easy replication of some parts of the circuit.

The Figure 7 shows the entire CIC filter implemented in Scilab. The four integrators were organized into four super blocks that can be replicated and saved as independent circuits, which allows the use in other projects. All $1/Z$ blocks need to be connected to clock type blocks, which adjust the time of the execution of the mathematical calculations. The clock blocks period was normalized to 1, and its startup time must

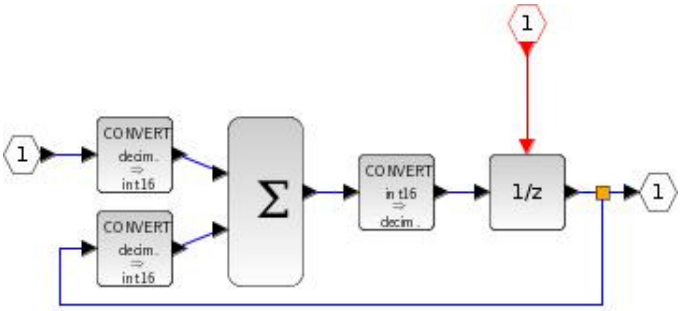

Fig. 6. Integrator super block

be 1 to compensate for the delay of the source, since it starts at -1 and we want it to start at 0.

B. Differentiator

The principle of the differentiator is the same as the integrator, but subtractors are used instead of adders, and the clocks have a period normalized of 16, due to the decimation factor of 16, and a delay of 2. It is important that the period of the clocks of all differentiators are synchronized so that no errors occur during the simulation. In the Figure 7 can be seen the whole CIC filter, where the 4 integrators are all connected to a single clock, as it makes it easier to adjust the period and delay. The same was done to the differentiators.

Unlike Matlab/Simulink, Scilab/Xcos does not work with discrete time by default, so when plotting the graphs there are some differences. To get around this problem in Scilab, when placing the scope we define it with a period smaller than the other clocks, making the signal appear to be discrete, than scope needs a unique clock.

In the figure 7, we can see the space savings provided by the super integrator blocks, note that the differentiators take up half of the entire circuit, as well as the ease of implementation, throughout the circuit, we notice the cascaded integrators, followed by the clock divider and derivatives. As a source we use the 'from workspace' block, that allows us to import data directly from the Scilab workspace. This data must be structured in two vectors, called time and values, respectively. Finally, to analyze the performance of the CIC filter, the output data was exported to the Scilab workspace with the block 'to workspace'.

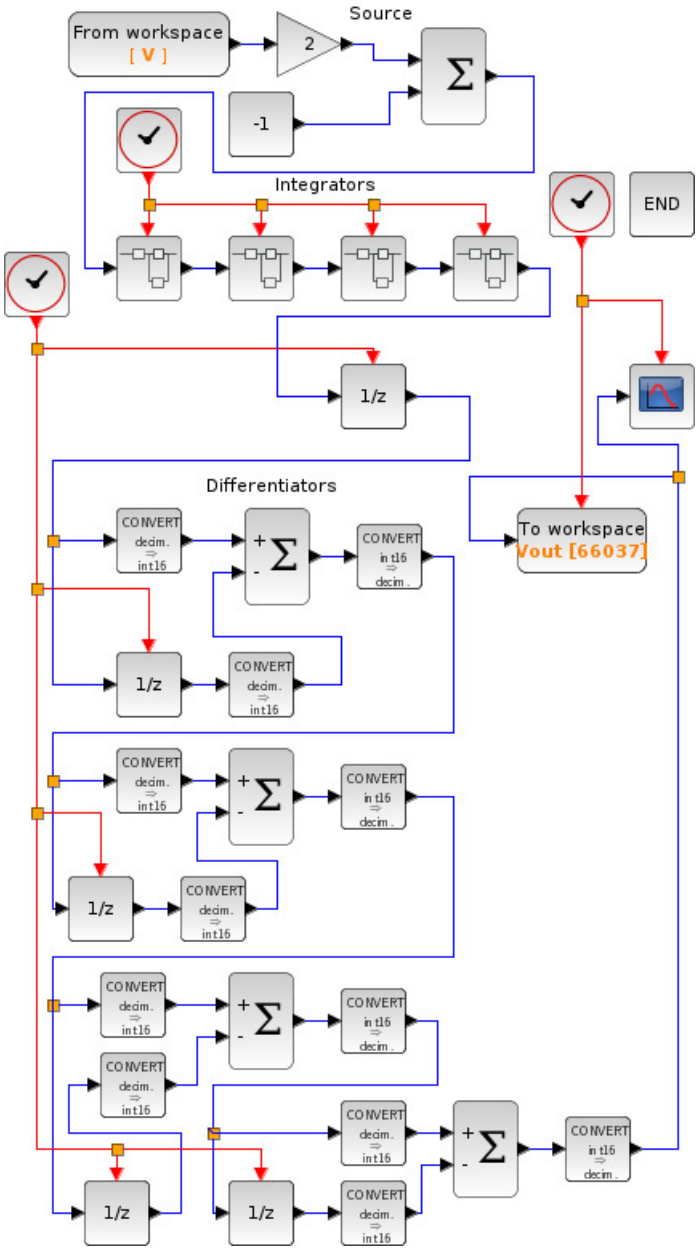

Fig. 7. CIC filter in Xcos/Scilab.

V. Simulation Results

This section presents the simulation results of the CIC filter implemented in Matlab/Simulink and Xcos/Scilab. Figure 8 presents the power spectrum density at the output of the CIC filter implemented in Xcos. It can be seen that the signal within the 20-kHz range remains unchanged when compared to the signal in Figure 2. On the other hand, the amplitude of the noise is attenuated by about 40 dB. The SNDR of this signal is 69.509 dB and the ENOB is 11.254 bits. It shows basically the same SNDR/ENOB as the modulator. It is worth mentioning that the FFT must be calculated considering a sampling frequency equal to 250 kHz due to the decimation rate of 16.

This output signal provides the exact same SNDR and ENOB of the CIC filter implemented in Simulink, discussed in the paper [8].

Figure 9 presents the reconstructed signal from the CIC filter implemented in Xcos. In this reconstruction a unitary voltage reference is considered. The least significant bit voltage is given by:

$$V_{LSB} = \frac{V_{REF}}{2^N} = \frac{1}{2^{16}} = 15.259\mu V, \qquad (2)$$

where V_{REF} is the reference voltage and N is the ADC

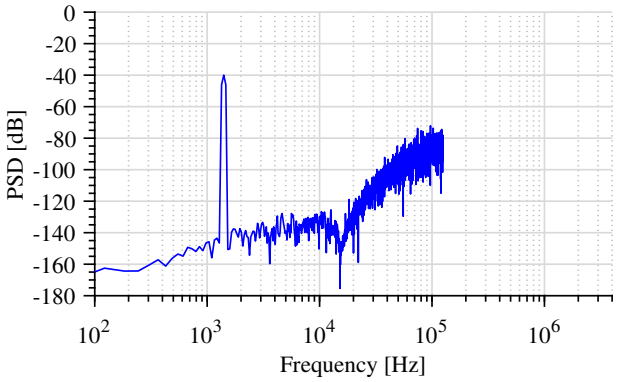

Fig. 8. Power spectrum density at the output of the CIC filter implemented in Xcos/Scilab for a 1.4038-kHz input signal with amplitude of -40 dBFS. 4,096-points FFT.

number of bits. The reconstructed signal has an amplitude of 10 mV. It is in agreement with the modulator input of -40 dBFS (amplitude of 10 mV). Also, it is possible to verify some high-frequency content in the signal waveform. It is related to the out-of-band noise generated by the noise-shaping effect. This high-frequency will be eliminated by FIR filters that will be placed after the CIC filter [7], which are the future steps of this work.

Note that the filter output shown in Figure 9, and the PSD illustrated in Figure 8 are identical to the figures obtained in the previous work [8]. The same results occur because the input stimuli are the same for comparison and validation purposes.

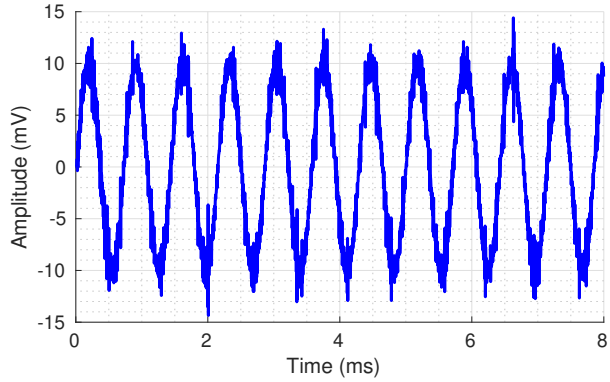

Fig. 9. Output signal from the CIC filter implemented in Scilab.

VI. Conclusions

This article aimed to show that it is possible to use free open software tools to develop and simulate digital filters. Having as main objective the re-implementation of a 4th-order CIC filter for an Audio Sigma-Delta ADC, previously developed in Matlab, in Scilab software. Both implementation results found were the same, showing that Scilab is a great tool to develop and simulate digital filters. The main implementation details are provided along the paper and are a basis to the implementation of different digital filters.

References

[1] F. Maloberti, *Data Converters*. Springer, 2011.
[2] R. Lv, W. Chen, and X. Liu, "A high-dynamic-range switched-capacitor sigma-delta ADC for digital micromechanical vibration gyroscopes," *Micromachines*, vol. 9, no. 8, 2018.
[3] A. Sukumaran and S. Pavan, "Low Power Design Techniques for Single-Bit Audio Continuous-Time Delta Sigma ADCs Using FIR Feedback," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 11, pp. 2515–2525, sep 2014.
[4] P. Malcovati and A. Baschirotto, "The evolution of integrated interfaces for MEMS microphones," *Micromachines*, vol. 9, no. 7, pp. 1–20, 2018.
[5] L. Sant, M. Fuldner, E. Bach, F. Conzatti, A. Caspani, R. Gaggl, A. Baschirotto, and A. Wiesbauer, "A 130dB SPL 72dB SNR MEMS microphone using a sealed-dual membrane transducer and a power-scaling read-out ASIC," *IEEE Sensors Journal*, vol. XX, no. XX, pp. 1–1, 2022.
[6] G. C. T. Richard Schreier, Shanthi Pavan, *Understanding Delta-Sigma Data Converters*, 2017.
[7] S. Parameswaran and N. Krishnapura, "A 100 μw decimator for a 16 bit 24 khz bandwidth audio modulator," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, 2010, pp. 2410–2413.
[8] E. Silva, N. Chagas, C. M. Muller, and P. Aguirre, "Design of a Digital CIC Filter for an Audio Sigma-Delta ADC," *Simpósio Sul de Microeletrônica*, 2022.