Comparative Analysis of Bandpass IIR Filters on Microphone PDM Signal for FPGA Implementation

Lukrecia Vulić, Ivan Aleksi, Tomislav Matić
Faculty of Electrical Engineering, Computer Science and Information Technology Osijek
Kneza Trpimira 2B, 31000 Osijek, Croatia
{lukrecija.vulic, ivan.aleksi, tomislav.matic}@ferit.hr

Abstract. *In this paper, a comparative analysis of quantisation effects on different types of Infinite Impulse Response (IIR) filters is performed. This research provides a basis for future applications in Field-Programmable Gate Array (FPGA) implementations. Data processing and filter evaluation are carried out using MATLAB. Butterworth, Chebyshev Type I, Chebyshev Type II and elliptical filters are selected as IIR filters. The experiments assess how different types of quantisation approximate the double-precision floating-point signal used as a reference. Several quantisation formats are considered, including Q7.24, Q2.16, and Q4.12. The metric used to compare quantisation levels and filter types is Mean Square Error (MSE) in combination with the Hamming window around frequencies of interest. The conclusions are based on the performance of the quantised version in comparison to the double-precision version and how well this format is suited to the intended hardware. The study finds that, for this scenario, the optimal ratio of resource consumption to minimum quantisation error occurs with a Butterworth filter at a quantisation level of Q2.16.*

Keywords. sound signal processing, MEMS microphone, PDM, IIR, quantisation, FPGA

1 Introduction

Digital filters are fundamental components in modern signal processing that enable the manipulation and analysis of discrete-time signals. They are used in various applications, including telecommunications, audio processing and biomedical engineering Oppenheim, 1999. In this paper, the processing of audio signals is performed using a Micro-Electro-Mechanical Systems (MEMS) microphone that outputs a pulse density modulation (PDM) digital signal, with implementation targeted on a Field-Programmable Gate Array (FPGA) platform. Due to the nature of the PDM format, suitable filtering is required to reconstruct the original signal. As the FPGA does not provide native floating-point support, the use of fixed-point arithmetic is required, which necessitates the quantisation of the filter parameters.

Infinite Impulse Response (IIR) filters are particularly appreciated because they achieve the desired frequency responses with fewer coefficients compared to their Finite Impulse Response (FIR) counterparts Hamming, 1989. Among the different IIR filter structures, Direct Form I is characterised by its straightforward implementation, where the filter's difference equation into a system diagram Oppenheim, 1999. This structure offers advantages in terms of simplicity and reduces the risk of numerical overflow in fixed-point arithmetic Oppenheim, 1999. When implementing Direct Form I filters on FPGAs to process PDM microphone signals, several aspects need to be considered, including resource utilisation, numerical stability and real-time performance.

This study examines the effect of quantisation on the following Direct Form I filter types: Butterworth, Chebyshev Type I, Chebyshev Type II and elliptical. The quantisation levels chosen reflect both common standards and forms that are suitable for the hardware to which they can be applied. The suitability of the individual quantisation forms is determined by comparing the double precision floating-point and fixed-point response to the step function and the response to the sample recording of an audio signal.

The rest of the paper is organised as follows. Section 2 gives an overview of related works, while section 3 describes the theoretical background of signal type, filter design and implementation form. Section 4 gives an overview of the methodology and Section 5 presents the experimental evaluation with the results and discussion. Conclusions and future remarks can be found in Section 6.

2 Related research

The comparative analysis of filters for specific applications is not new. In Ramdoss and Pl, 2013, for example, the authors conducted a comparative study to determine the best filtering method for audio signal processing, more specifically speech signal processing. In this work, the authors evaluated the effectiveness of each filter by observing how well chosen filters perform con-

tinuous speech segmentation. Filtering this type of signal allows for the removal of unwanted background noises and other interferences. The experiments were conducted using the OGI-MLTS corpus Muthusamy et al., 1992 and a sample voice signal to evaluate the performance of the different filters.

In Carletta et al., 2003 a framework for the implementation of IIR filters in fixed-point arithmetic on an FPGA was developed. The aim of the paper was to determine the bit size of the filter coefficients while maintaining accurate performance. The methodology is used in a magnetic bearing control system where a compensator is generated according to the selected bit width. In that way, the authors ensured that there is no overflow in registers. It was also shown that the approach creates a balance between computational accuracy and efficient utilisation of hardware resources.

The authors in Zelmat et al., 2023 address both IIR filter design and precision by applying multiobjective optimisation for both aspects. Similarly, Hormigo and Caffarena, 2021 proposes an emulator to accelerate word length optimisation in FPGAs. The method requires the user to reconfigure the FPGA only once, incorporating precision limiters into the design itself. In that way, it is possible to test different word length combinations on the circuit without the need for reconfiguration.

The implementation of IIR filters in fixed-point arithmetic is considered in Pinheiro et al., 2010. In the research, filter types such as: Butterworth, Chebyshev Type I and II and elliptical are evaluated based on the quality factor parameters and the influence of cut-off frequencies on the number of significant bits needed to represent the coefficients. It becomes clear that Direct Form I structures are preferable for FPGA implementation as they have lower latency and handle parallel operations better. The results also show that most filters achieve acceptable performance with only 10–12 coefficient bits, making them suitable for FPGAs with limited resources.

These studies collectively demonstrate that the choice of filter type, coefficient precision, and implementation structure significantly affects both performance and hardware efficiency in fixed-point FPGA designs. Building on this foundation, the present work examines the viability and challenges of implementing IIR filters in fixed-point arithmetic for FPGA applications, with an emphasis on minimising quantisation error. This approach enables the identification of filter–structure combinations that achieve the best trade-off between numerical accuracy and hardware efficiency.

3 Theoretical background

3.1 Pulse density modulation

Literature describes PDM as a 1-bit digital audio signal. A digital microphone captures this type of signal. This step is done via an analog preamplifier and a PDM modulator. The analog signal is initially captured by a microphone component and immediately amplified by the preamplifier. The signal is then sampled at a high frequency and quantised in the PDM modulator. The modulator generates a single-bit output signal that has also undergone a noise shaping process. Finally, the digital microphone interface receives the master clock and passes on the sampled bit stream. The device using the digital microphone provides the master clock, the rate of which is determined by the sampling rate of the system and the bit transmission rate Evans, n.d.

3.2 Filter Design

The choice of digital filter type is crucial for any signal processing task. This choice is limited by the given task and by the computing power and resources available for the task, in this case an FPGA board. There are two major divisions of digital filters: the non-recursive filters and the recursive filters. Non-recursive filters are commonly referred to as FIR filters and recursive filters are known as IIR filters Hamming, 1989. FIR filters are not recursive and therefore only use the previous and current inputs when calculating the filter output. On the other hand, IIR filters take current and past inputs as well as the past outputs when calculating the filter output. This distinction between the filters leads to several notable differences. The loop architecture of IIR filters requires fewer coefficients for sharper transitions, in contrast to FIR filters, which require a higher order of coefficients to achieve similar results National Instruments, 2023a. This indicates that IIR has superior computational performance and resource efficiency. The recursive properties of IIR can lead to potential drawbacks. If not constructed correctly, they can become unstable, whereas FIR structures maintain inherent stability Oppenheim, 1999.

Considering all these aspects, the selected filter type for this paper is the IIR filter, due to the limitations of the hardware platform planned for the implementation. The aim of this paper is to find the optimal IIR version for implementation on the FPGA platform. This paper focuses on the following IIR filter coefficient-approximation methods:

1. Butterworth approximation

   Butterworth filters are particular types of digital filters that are characterised by a flat, magnitude response in the passband. This characteristic requires a compromise characterised by a wide transition bandwidth, which is poor compared to other classical approximations. To achieve a narrow

transition bandwidth, a higher order of the Butterworth filter is required Lai, 2003.

2. Chebyshev approximation

   Chebyshev filters are calculated using Chebyshev polynomials, which are defined by the filter order. Chebyshev polynomials are used to shape the filter frequency response to minimise the maximum deviation of the passband. The error curve generated by this approximation has equal waves. Chebyshev filters are divided into two categories: Type 1 and Type 2 Lai, 2003. The Chebyshev Type 1 filter permits a ripple in the passband, whereas the Type 2 filter permits a ripple in the stopband Hamming, 1989.

3. Elliptic function

   Eliptical filters are characterised by the fact that they have ripples in both the passband and stopband. Their design is based on the theory of eliptical functions Hamming, 1989.

3.3 Direct Form I

Direct Form I is a digital filter implementation technique for IIR filter realization. Other common forms include Direct Form II, cascade form, parallel form, and lattice-based structures, each offering different trade-offs in memory usage, numerical properties, and suitability for fixed-point or floating-point implementation. The Direct Form I structure can be described as two FIR filters, one forward and one reversed. The forward FIR filter comprises numerator or *b* coefficients, and the reversed one uses denominator or *a* coefficients Inc., 2008. Its most valuable attribute is found in fixed-point arithmetic, where there is no possibility of filtering internal register overflow. This advantage is due to the single summation point in the filter structure, as well as the characteristic fixed-point wraparound behavior. One notable downside, compared to other types of implementation, is the double amount of delay elements required Smith, 2007. Another important consideration is the possibility of filter stability being affected by roundoffs needed for fixed-point arithmetic. This issue is solved by splitting the filter's transfer function into second-order sections (SOS) or biquads whose structure is shown in Fig.1 Smith, 2007. The biquads can be arranged in a cascaded and parallel structure. For the purposes of this paper, a cascaded structure is observed. These structures are used for larger filter orders and represent a product of the number of biquads Inc., 2008. The SOS bit-width parameter between each SOS has to be adjusted to handle the influence of input and output gains. These widths have to be adjusted in order not to lose data and retain a level of precision Mauerer, 2018.

3.4 Qm.n quantisation format

In digital signal processing, numerical data is usually represented in either floating-point or fixed-point formats. This choice depends on the constraints of the application, such as accuracy, dynamic range and hardware resources. Floating-point representation is defined by the IEEE 754 standard and represents numbers with one sign bit, one exponent and mantissa "IEEE Standard for Floating-Point Arithmetic", 2019. This enables a large dynamic range and high accuracy, making it suitable for general calculations Oppenheim, 1999. In resource-constrained environments, however, a fixed-point representation is used due to the lower hardware complexity and deterministic performance Meyer-Baese, 2014. The $Qm.n$ format is a fixed-point convention that divides a binary word into m integer bits containing a sign and n decimal bits. This enables efficient encoding of real numbers without the need for floating-point units. Converting a floating-point number to $Qm.n$ format involves scaling the value by 2^n, then rounding or truncating to the nearest integer and implementing overflow or saturation management to maintain numerical integrity Texas Instruments, 2001. This conversion reduces numerical accuracy and dynamic range while significantly increasing execution speed and resource efficiency, making Q-format arithmetic particularly advantageous in applications such as audio processing and real-time control Widrow and Kollár, 2008.

4 Methodology

The methodology in this paper focuses on evaluating the impact of the quantisation process on filter performances, focusing on digital audio signals. A key challenge in this process is that the target FPGA lacks native support for floating-point arithmetic, which makes it incompatible with the floating-point filter coefficients generated by MATLAB. Furthermore, the optimal filter type and corresponding quantisation format is not immediately obvious. To overcome these limitations,

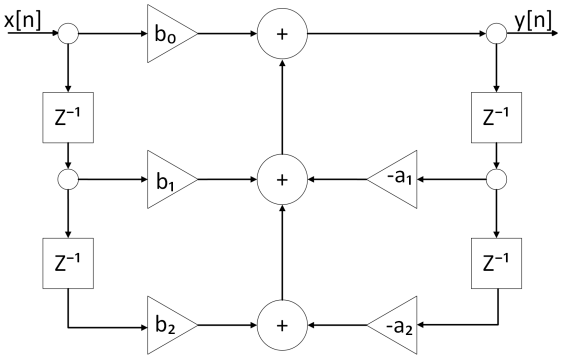

Figure 1. Block diagram of a Direct Form I IIR filter SOS National Instruments, 2023b

the filters are first designed in MATLAB using the Direct Form I structure. These filters are then converted into a fixed-point representation using a Python-based tool chosen for its flexibility and open-source availability. A simulation framework Mauerer, 2018 in Python is then used to evaluate the impact of the quantisation process on filter accuracy compared to the one in double precision. This evaluation is performed using a step function and a PDM audio recording captured by an FPGA device.

4.1 Sound production

The test samples are designed to mimic the real conditions that the planned system would encounter. The sound samples are generated artificially, using pre-recorded ambient noise and a sine wave. Human, non-speech sounds and outdoor urban noises were selected as background sounds. A bursting sine wave is a sine wave that occurs for a certain number of periods and then remains inactive. The frequency of the sine wave is set to 11.025 kHz due to the limited range of available MEMS microphones Analog Devices, n.d. and the value of the sound sampling frequency in MATLAB. When generating or analysing a sinusoidal signal in digital systems, it necessary the sine wave frequency is an integer divisor of the sampling frequency Chassaing and Reay, 2011. This exact periodicity ensures that the sampled signal repeats perfectly and without phase drift, resulting in a clean frequency domain analysis without spectral leakage in the FFT. For example, using a sampling frequency of 44.1 kHz and a sine wave frequency of 11.025 kHz results in exactly four samples per sine wave period. The frequency range of this artificial sound can be seen in Fig.2.

The dataset used for the environmental sounds is named ESC-50 and can be found in Piczak, 2015. These test samples may be used as individual sounds and as combinations. In contrast to the dataset, these sounds were cut to a duration of 4 seconds due to hardware limitations of the FPGA. The BRAM memory of the selected FPGA device can only hold up to 4 seconds of samples.

4.2 Hardware setup

The test setup consists of a loudspeaker and a MEMS microphone connected to an FPGA. The loudspeaker is used to reproduce ambient sounds and the bursting sine wave signal. The FPGA board used is the Nexys A7 from Digilent Digilent Inc., n.d. and the onboard MEMS microphone is used Analog Devices, n.d.

The FPGA is programmed to capture data from the MEMS microphone at a rate of 1 MHz and store a 4-second PDM sample in the block RAM memory. This sample is later transferred to a PC via a UART connection. This process of recording PDM data on an FPGA is designed as a Finite State Machine (FSM), see Fig.3.

The FSM is designed to record a number of N samples, each of which is stored in memory as soon as it is recorded. After N samples have been recorded and stored in memory, they are read from memory one by one and sent via a UART connection as long as there are samples to be read from memory and a UART connection is available.

4.3 Evaluation procedure

The evaluation procedure is separated into a number of steps. Firstly, the focus is on choosing the correct filter type as well as making sure that the filter is stable and is of minimum order. This is done through a Signal Processing Toolbox in MATLAB MathWorks, Inc., 2025. The filters are represented in Direct Form I structure. The generated parameters are exported to an open-source tool that is specialised for filter coefficient quantisation and testing filters with quantised co-

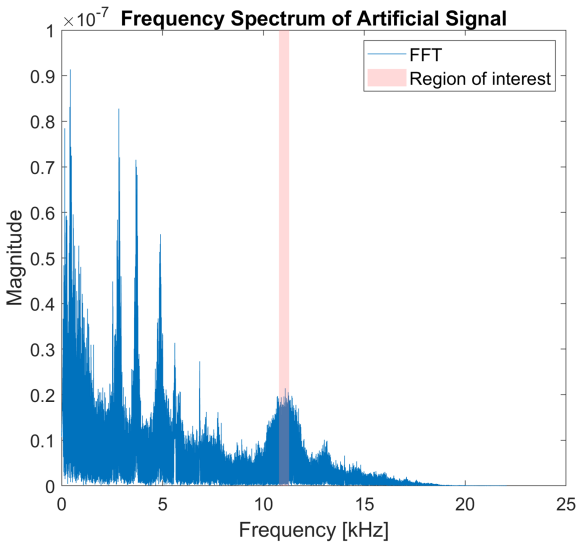

Figure 2. FFT analysis of the test sound used for the sine extraction at 11.025 kHz with a sampling frequency of 44.1 kHz

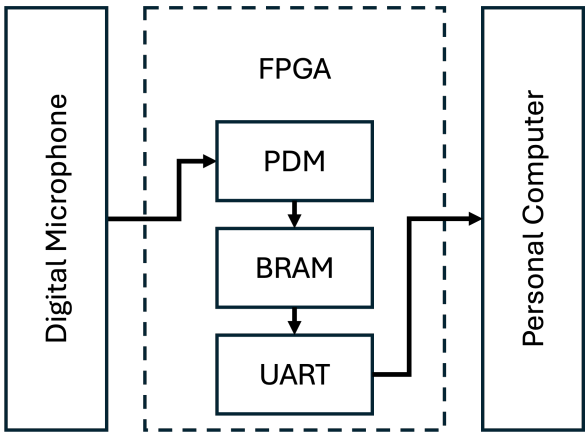

Figure 3. Digital MEMS microphone and PDM signal processing chain on an FPGA board using FSM

efficients. The stability of the quantised coefficients is determined based on the pole position in the complex plane. The next step is focused on observing the double precision floating-point and quantised filter's response to a step function. The unit step function is a signal that has a constant value of zero for all time before a specified starting point, and a constant value of one for all time after and including that point Shmaliy, 2007. Using this activation signal the filter type and quantisation settings can be vetted through observing the error rates between the quantised and floating-point representation responses. Finally, the filter is tested on the PDM signal described in subsection 4.1. The validity of the filter is vetted by observing how it filters out a PDM audio recording with a known frequency included. This recording is filtered and processed by Fast Fourier Transform (FFT).

The steps are numbered and explicitly stated in the following paragraph:

1. designing the filter using MATLAB with desired specifications
2. exporting filter coefficients to an open-source Python tool VIIRF Mauerer, 2018
3. quantise filter coefficients to the chosen fixed-point format and confirm filter stability in this format
4. setting the value of the SOS bit size
5. simulate and evaluate the filter using test signals
   (a) compute MSE of step response
   (b) compute MSE of FFT of a PDM signal.

5 Experimental evaluation

5.1 Observed parameters

Four filter types were chosen for this evaluation process: eliptical, Butterworth, Chebyshev Type I and Chebyshev Type II. These filters were chosen due to their popularity and their varying effects as in Pinheiro et al., 2010. A bandpass filter was chosen with a center frequency of 11.025 kHz and a bandwidth of 250 Hz on each side, resulting in a total passband ranging from 10.775 kHz to 11.275 kHz. The *Apass* parameter represents the maximum allowed variation in the passband Proakis and Manolakis, 2006, the chosen values for this parameter can be seen in Table 1.

It was also important to look at how each filter reacts to different quantisation parameters, as they all have varying tolerances for this process. As mentioned, this comparative analysis is used to determine which filter would best be suited for FPGA fixed-point implementation.

Calculations on the FPGA are done using Digital Signal Processing (DSP) slices, which have a certain data width that they handle. In cases where this width is exceeded, calculations have to be transferred to a new slice. It is important to keep calculations in a certain bit range to minimize the use of the available DSP slices. The SOS bit-size parameter refers to the register size in between SOS of the filter, where the output of each section in stored and used for further calculations. This parameter depends on the size of filter input, the value of the gain and the user's input on additional bits. Depending on the gain value for each section there may be need for larger register size at the section output. The user can also impact this parameter if there is need for extremely precise calculations.

Errors in the quantisation process are also observed, primarily in the Mean Square Error (MSE) format after testing the filters. The MSE is calculated based on a fixed and double precision floating-point step response, as well as the Fourier analyses of these two formats. The floating-point value is taken as a reference point to see how the conversion and round off error is manifested in fixed-point format response. A Hamming window is applied and centered on the bandpass region for the MSE FFT calculation. This approach prioritises accuracy in the passband by assigning higher weights to the error in that region while suppressing the contributions outside of it. Stability and usability of the filter after the quantisation process were also looked at in terms of pole placement in a complex plane and the value of the filter output. The parameters used in this study are presented in Table 1. The *Apass* values were selected to investigate the impact of varying passband ripple levels on quantisation effects. The effects of different *Apass* values are shown in Fig.4. The Q2.16, Q4.12, and Q7.24 formats were chosen to align with common register widths in fixed-point hardware implementations, thereby enabling a comparative evaluation of their performance. The SOS bit-width values were varied between 17 and 28 bits to examine how the precision of intermediate results influences the accuracy of the final output.

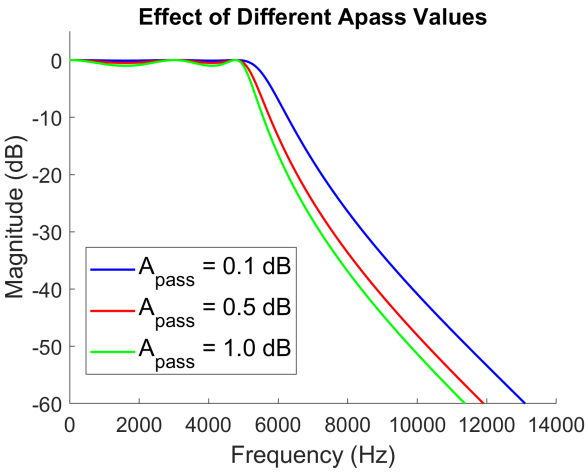

Figure 4. Example of the effects of different Apass values

Table 1. EXPERIMENT PARAMETERS

Apass [dB]	Q format	SOS bit-size
0.10	Q2.16	17–28
0.25	Q4.12	17–28
0.50	Q7.24	17–28

5.2 Results and discussion

The results are obtained iteratively by repeating the procedure described in subsection 4.3 for each parameter combination. Fig. 5(a) shows the effects of changing the parameters of the Butterworth filter. It is immediately apparent that the calculated error for the FFT MSE of the Q4.12 format is higher compared to others for all Apass values, and they are represented by the color green. All step response errors approach zero which indicates a good fixed-point approximation. The FFT MSE for Q2.16 and Q7.24 also approaches zero as the SOS bit size increases for all Apass values.

The graph for Chebyshev Type I filter, as seen in Fig. 5(b), has different scaling of the x axis. This is due to the need for accommodating larger values for cases when the output gain is lesser than one Mauerer, 2018. From the graph it is evident that values on the step response MSE are approaching zero. The FFT MSE values are reducing as the SOS bit size goes up, in all cases but one. For the Q7.24 format this value approaches zero, while Q2.16 preforms with somewhat lesser precision, its stagnant values being as high as 0.1. The worst performing format is once again Q4.12 as its error rate ranges from 0.3 to 0.6.

Similar results are noticed in Fig. 5(c) for Chebyshev Type II filter design. Step responses for all Q formats approach zero. On the other hand, the MSE for the FFT depicts a significant error. Despite the increase of SOS bit size, the MSE value stagnates in range of 0.1 to 0.2 for Q2.16 and Q7.24 formats. The Q4.12 format gives subpar values compared to other formats.

The elliptical filter design, as seen in Fig. 5(d), produces such filters that SOS bit size has no impact on the MSE of any Q formats. The values are constant and it is yet again apparent that calculations using Q4.12 format produce higher error rates. The error in the step response once again approaches zero, and the FFT error rate for the other Q formats for all Apass values is less than 0.1.

For the final part of this analysis, the viability of each filter type and Q format is observed. As previously mentioned, it is important to take into consideration both the error of calculation and the constrictions of the FPGA. As mentioned already, the targeted FPGA DSP slices allow calculations in the format of 25x18 bits and therefore to achieve minimal resource usage it is imperative to limit the SOS bit size parameter to be up to 25 bits in length, and coefficient size to be up to 18 bits. The SOS bit size across all filter types is seen in Table 1. The Q formats overall bit size is set to be either 16, 18, or 31 in size.

Table 2 shows the best results obtained thru this experiment. For each of the four filter types the best result of each Q format with the corresponding SOS bit size is extracted, the best result being a minimal error of the FFT comparison between double precision and fixed-point. MSE of step response is also included in Table 2 and shows that the minimal error for most cases is achieved with Q7.24 format. In this table the best results are bolded and it is clear that for three types of filters the smallest error for FFT is produced using Q7.24 format which is to be expected. With an extended fraction bit size precision in calculation is enhanced and a better approximation of the floating-point format is achieved. Upon further review, Q2.16 format appears to be a clear second choice. The clear outlier remains the Q4.12 format. The SOS bit size in t Table 2 is within the bounds of 25 bits and therefore does not impact the choice of filter and format.

Taking into consideration the FPGA constrains mentioned and the MSE of the FFT analysis and step response chosen as a metric, it is therefore concluded that the best approach to implementation is to use the Butterworth filter with the Q2.16 format. This Q format would ensure that a minimal number of DSP slices is utilised, as the SOS bit width is also in the correct bound. The MSE metric is also the smallest of any in this format.

Table 2. MSE EXPERIMENTAL RESULTS

Filter	Apass	Q format	SOS bit size	MSE step	MSE FFT
Butterworth	0.10	Q2.16	22	3.45e-08	3.57e-03
	0.25	Q4.12	22	5.46e-06	2.14e-01
	0.50	Q7.24	22	**4.37e-10**	**7.58e-07**
Chebyshev I	0.10	Q2.16	23	1.54e-07	1.69e-02
	0.10	Q4.12	22	2.17e-06	3.40e-01
	0.50	Q7.24	23	**3.51e-10**	**5.40e-07**
Chebyshev II	0.50	Q2.16	19	**2.12e-06**	1.20e-01
	0.50	Q4.12	18	5.14e-06	1.96e-01
	0.50	Q7.24	18	2.56e-06	**1.19e-01**
Elliptic	0.25	Q2.16	17	**4.93e-07**	3.18e-02
	0.50	Q4.12	17	1.63e-06	2.66e-01
	0.50	Q7.24	17	2.10e-06	**1.86e-02**

6 Conclusion

The paper demonstrates the viability and challenges of choosing an implementation format of IIR filters – Butterworth, Chebyshev Type I and II, and elliptical – in fixed-point arithmetic for FPGA applications. The design of the filters in floating-point and its validity was verified using MATLAB, while the effects of quantisation on filter stability and errors in approximation caused by this process were observed in order to determine the best-suited combination for FPGA implementation. The specific FPGA implementation sup-

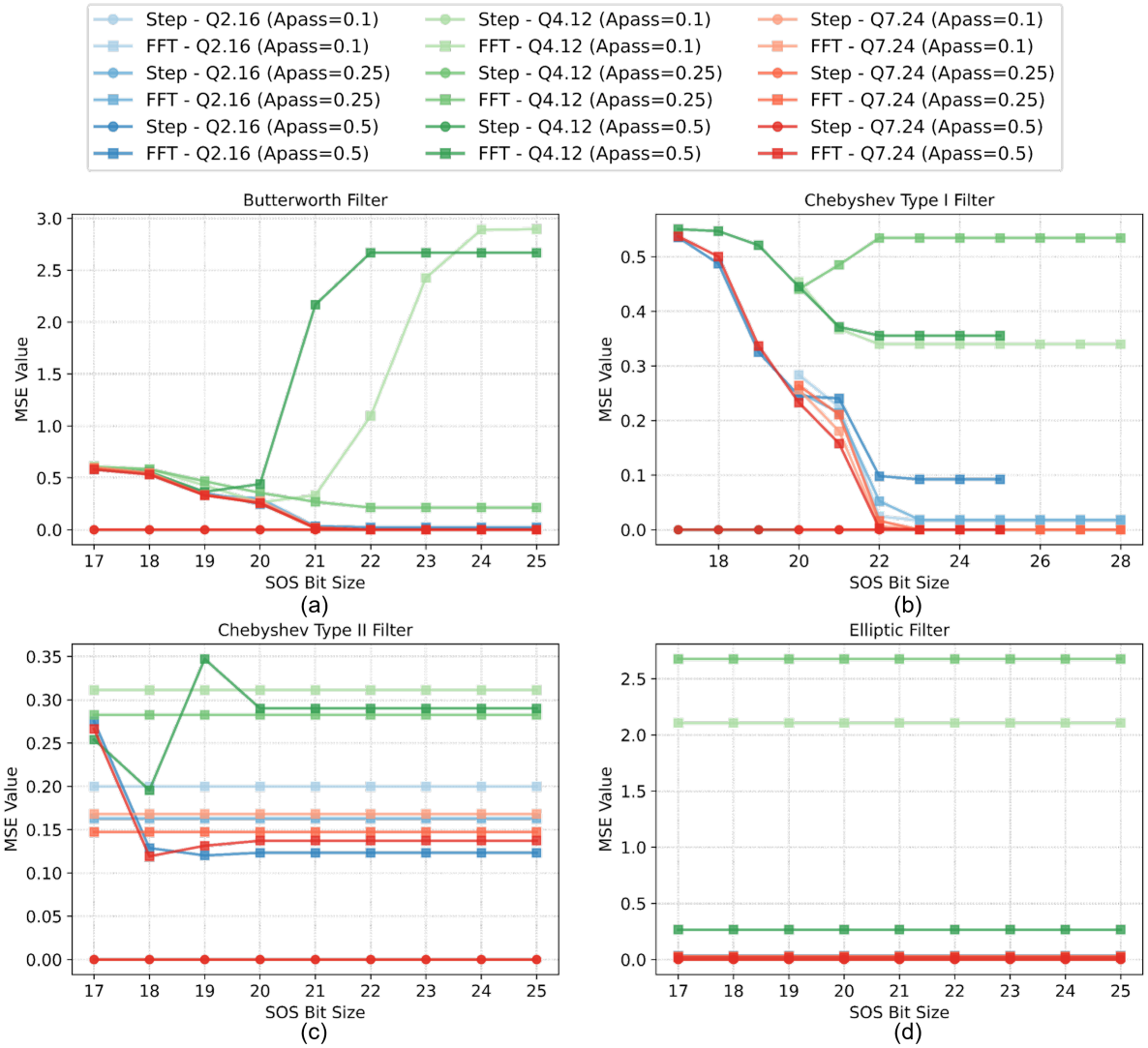

Figure 5. Comparison of different filter types and quantisation formats (a) Butterworth filter (b) Chebyshev Type I filter (c) Chebyshev Type II filter (d) Eliptical filter

posed is Direct Form I, which is a straightforward IIR implementation using SOS that additionally reduces the effects of quantisation. The coefficient stability is checked after the quantisation takes place, after which both quantised and floating-point coefficient are used to filter a step signal and a audio recording. In this way quantisation responses are measured up to the reference captured by the floating-point responses.

All the filter types and corresponding quantisation levels have produced stable filters. Regarding the comparison, across most filter types the minimal quantisation error was produced using the Q7.24 format. However, taking into consideration the resource management needed for low-level implementation, the authors conclude that the best ratio of error management and resource consumption is achieved with the Q2.16 format. The minimum error for this format is achieved with the Butterworth filter type.

This research opens up several directions for future investigation. The obvious one is the implementation and real-time testing of the designed filters on the FPGA platform, using VHDL to verify synthesis, timing and resource utilisation. Future work could also incorporate dynamic range analysis and noise sensitivity, especially for audio and sensor applications. Finally, extending the analysis to adaptive filtering algorithms or multi-channel systems, such as beamforming arrays is also a possibility.

Acknowledgement

The authors would like to thank Infineon Technologies Austria AG for their support in this research in the framework of the Important Project of Common European Interest (IPCEI) PhD program on Microelectronics.

References

Oppenheim, A. (1999). *Discrete-time signal processing*. Pearson Education. https://books.google.hr/books?id=geTn5W47KEsC

Hamming, R. (1989). *Digital filters (3rd ed.)* Prentice Hall International (UK) Ltd.

Ramdoss, A., & Pl, C. (2013). A comparative study on various types of filters in audio signal processing.

Muthusamy, Y. K., Cole, R. A., & Oshika, B. T. (1992). The ogi multi-language telephone speech corpus. *Proceedings of the 2nd International Conference on Spoken Language Processing (ICSLP 1992)*, 895–898. https://doi.org/10.21437/ICSLP.1992-276

Carletta, J., Veillette, R., Krach, F., & Fang, Z. (2003). Determining appropriate precisions for signals in fixed-point iir filters. *Proceedings of the 40th Design Automation Conference (DAC)*, 656–661. https://www.cecs.uci.edu/~papers/compendium94-03/papers/2003/dac03/pdffiles/38_3.pdf

Zelmat, M., Lamini, E.-S., Tagzout, S., Belbachir, H., & Belouchrani, A. (2023). Multi-objective approach for iir filter design and bit-width optimization. *Circuits, Systems, and Signal Processing*, *42*(8), 4836–4867. https://doi.org/10.1007/s00034-023-02334-1

Hormigo, J., & Caffarena, G. (2021). Fpga acceleration of bit-true simulations for word-length optimization. *2021 IEEE 28th Symposium on Computer Arithmetic (ARITH)*, 119–122. https://doi.org/10.1109/ARITH51176.2021.00033

Pinheiro, E., Postolache, O., & Silva Girão, P. (2010). Fixed-point implementation of infinite impulse response notch filters. *Metrology and Measurement Systems*, *17*. https://doi.org/10.2478/v10178-010-0019-3

Evans, B. L. (n.d.). Understanding pdm digital audio [Course materials, The University of Texas at Austin, Accessed: May 16, 2025].

National Instruments. (2023a). Fir and iir filters [Updated 2023-02-21; accessed 2025-09-05].

Lai, E. (2003). 7 - infinite impulse response (iir) filter design. In E. Lai (Ed.), *Practical digital signal processing* (pp. 145–170). Newnes. https://doi.org/https://doi.org/10.1016/B978-075065798-3/50007-2

Inc., X. (2008, March). Infinite impulse response filter structures in xilinx fpgas. https://www.xilinx.com/support/documentation/white_papers/wp330.pdf

Smith, J. O. (2007). *Introduction to digital filters with audio applications* [online book].

Mauerer, M. (2018). Viirf: Versatile iir filter implementation [Accessed: 2025-04-17]. https://github.com/MauererM/VIIRF

National Instruments. (2023b). IIR SOS Specifications - LabVIEW Digital Filter Design Toolkit API Reference [Accessed: 2025-05-26]. https://www.ni.com/docs/en-US/bundle/labview-digital-filter-design-toolkit-api-ref/page/lvdfdtconcepts/iir_sos_specs.html

Ieee standard for floating-point arithmetic. (2019). *IEEE Std 754-2019 (Revision of IEEE 754-2008)*, 1–84. https://doi.org/10.1109/IEEESTD.2019.8766229

Meyer-Baese, U. (2014). *Digital signal processing with field programmable gate arrays* (4th ed.). Springer. https://doi.org/10.1007/978-3-319-00584-1

Texas Instruments. (2001). *Tms320c55x dsp programmer's guide* [Literature Number: SPRU376B]. Texas Instruments. Dallas, TX. https://www.ti.com/lit/ug/spru376a/spru376a.pdf?ts=1747302209728&ref_url=https%253A%252F%252Fwww.google.com%252F

Widrow, B., & Kollár, I. (2008). *Quantization noise: Roundoff error in digital computation, signal processing, control, and communications*. Cambridge University Press.

Analog Devices. (n.d.). ADMP421: High-performance, low power, digital output MEMS microphone [Datasheet, Accessed: May 16, 2025]. https://www.analog.com/media/en/technical-documentation/obsolete-data-sheets/ADMP421.pdf

Chassaing, R., & Reay, D. (2011). *Digital signal processing and applications with the tms320c6713 and tms320c6416 dsk*. Wiley. https://books.google.hr/books?id=LO_whVmhXE8C

Piczak, K. J. (2015). ESC: Dataset for Environmental Sound Classification. *Proceedings of the 23rd Annual ACM Conference on Multimedia*, 1015–1018. https://doi.org/10.1145/2733373.2806390

Digilent Inc. (n.d.). *Nexys a7 reference manual* [Accessed: May 16, 2025]. https://digilent.com/reference/programmable-logic/nexys-a7/reference-manual

MathWorks, Inc. (2025). Digital and analog filters [Accessed: 2025-09-05]. https://www.mathworks.com/help/signal/digital-and-analog-filters.html

Shmaliy, Y. (2007, September). *Continuous-time systems*. https://doi.org/10.1007/978-3-642-14325-0_1

Proakis, J. G., & Manolakis, D. G. (2006). *Digital signal processing: Principles, algorithms, and applications* (4th). Pearson Prentice Hall.