IP Generation for an FPGA-based Audio DAC Sigma-Delta Converter

Ralf Ludewig[1], Oliver Soffke[1], Peter Zipf[1], Manfred Glesner[1],
Kong Pang Pun[2], Kuen Hung Tsoi[2], Kin Hong Lee[2], and Philip Leong[2]

[1] Institute of Microelectronic Systems
Darmstadt University of Technology, Germany.
{ludewig,soffke,zipf,glesner}@mes.tu-darmstadt.de
and
[2] Department of Computer Science and Engineering
Chinese University of Hong Kong, Shatin NT HK.
{kppun,khtsoi,khlee,phwl}@cse.cuhk.edu.hk

Abstract. In this paper we describe a parameterizable FPGA-based implementation of a sigma-delta converter used in a 96kHz audio DAC. From specifications of the converter's input bitwidth and data sampling frequency, VHDL generic parameters are used to automatically generate the required design. The resulting implementation is optimized to use the minimum internal wordlength and number of stages. We prototyped the converter on an FPGA board for verification purposes and the results are presented.

1 Introduction

With recent improvements in the density of field programmable gate array (FPGA) devices, systems with an increasingly higher level of integration are possible. Digital signal processing is an important application area for FPGAs and such systems often require data converters to provide analog outputs from digital domain representations. Sigma-delta modulator based digital to analogue converters (DAC) can be efficiently implemented on FPGA devices since they are entirely digital. Including such converters in the FPGA provides a high degree of flexibility and reduces costs and time to market. Moreover, it becomes possible to develop single chip, customized converters which are not available commercially, e.g. an application may require a mix of different converters of different orders operating at different frequencies. Including such converters into the FPGA provides a high degree of flexibility and reduces costs and time-to-market. In this paper, a flexible sigma-delta converter for an audio frequency DAC as proposed in [1] is described. It is modeled as a parameterizable IP core description which can be used to generate the actual implementation of converters accepting 16-, 20-, and 24-bit input values at data sampling rates between 32 and 96 kHz. Compared with the design in [1] which was made in Handel-C, the IP core presented in this paper is more modular, automatically determines the required internal bitwidths to avoid overflow, is written entirely in VHDL and is approximately two times faster yet occupies less resources.

The paper is structured as follows: Section 2 describes the basic principle of operation of $\Sigma\Delta$-converters. The architecture of the implemented DAC is discussed in Section 3. Section 4 deals with the system-level simulation of the model and in Section 5 the experimental results of an implemented prototype are shown. Finally, we end up with some conclusions in Section 6.

2 Basics of $\Sigma\Delta$-Converters

The basic task of a $\Sigma\Delta$-converter is to quantize the input signal, which can be either continuous or discrete in time. This applies for both the analog-to-digital and the digital-to-analog conversions. 1-bit quantization is most popular for simple circuit design [2]. The one-bit signal is perfectly suited as the input or output of a digital system.

The quantization with one bit resolution can be achieved without significant quality loss, by oversampling the signal by a factor of M and shifting the quantization noise to high frequencies where no signal frequency components are present. This is usually referred to as *noise shaping*.

The signal and the noise can then be separated with a lowpass filter of the appropriate order. This lowpass works in the digital domain for analog-to-digital converters and in the analog domain for digital-to-analog converters. As this paper deals with a digital-to-analog converter we will focus our considerations on this type, but the basic theory can be applied for both types.

Figure 1 shows a simple $\Sigma\Delta$-converter which accepts a discrete time, continuous value input signal and outputs a discrete time, discrete value output signal with one bit resolution shifting the quantization noise to high frequencies. For the analysis of this $\Sigma\Delta$-converter a linearized model can be constructed as shown in Figure 1(b) by replacing the quantizer by an injection of additive noise. The output $Y(z)$ is the superposition of the input signal $X(z)$ transformed by the system and the noise signal $N(z)$ also transformed by the system:

$$Y(z) = H_x(z)X(z) + H_n(z)N(z)\,, \tag{1}$$

where $H_x(z)$ denotes the signal transfer function and $H_n(z)$ denotes the noise transfer function, which can be derived from

$$H_x(z) = \left.\frac{Y(z)}{X(z)}\right|_{N(z)=0} \tag{2}$$

$$H_n(z) = \left.\frac{Y(z)}{N(z)}\right|_{X(z)=0} \tag{3}$$

This is depicted in Figure 2. From figure 2a we find that

$$Y(z) = H(z)\Big(X(z) - Y(z)\Big)\,, \tag{4}$$

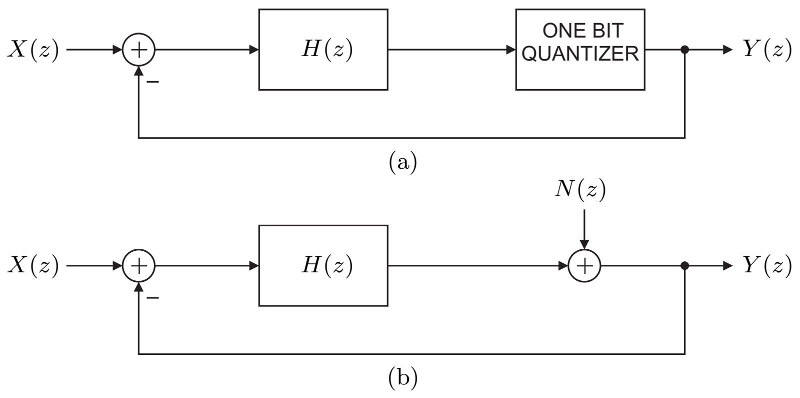


Fig. 1. Simple discrete time $\Sigma\Delta$-converter quantizing continuous value input signals with one bit resolution (a) and a system theoretic model of it with the quantizer replaced by additive noise injection (b).

which finally yields the signal transfer function $H_x(z)$:

$$H_x(z) = \frac{Y(z)}{X(z)} = \frac{H(z)}{1 + H(z)} \,. \tag{5}$$

Using the same procedure we also find the noise transfer function $H_n(z)$ from figure 2b:

$$Y(z) = N(z) - H(z)Y(z) \tag{6}$$

$$\Rightarrow H_n(z) = \frac{Y(z)}{N(z)} = \frac{1}{1 + H(z)} \,. \tag{7}$$

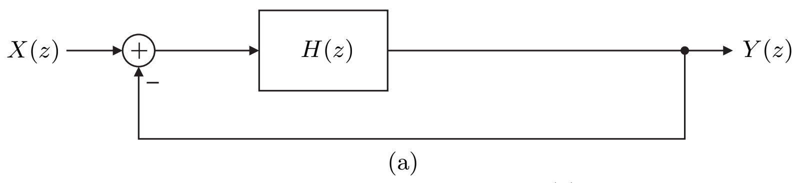
An integrator is used for $H(z)$ to implement the first order noise shaping. To avoid an algebraic loop due to the feedback in the $\Sigma\Delta$-converter, the integrator is chosen to have no direct feedthrough. Thus, it can be described in the time domain by

$$y_n = y_{n-1} + x_{n-1} \,. \tag{8}$$

Transforming this difference equation in the z-domain yields the transfer function $H(z)$:

$$Y(z) = z^{-1}Y(z) + z^{-1}X(z) \tag{9}$$

$$\Rightarrow H(z) = \frac{Y(z)}{X(z)} = \frac{z^{-1}}{1 - z^{-1}} \,. \tag{10}$$



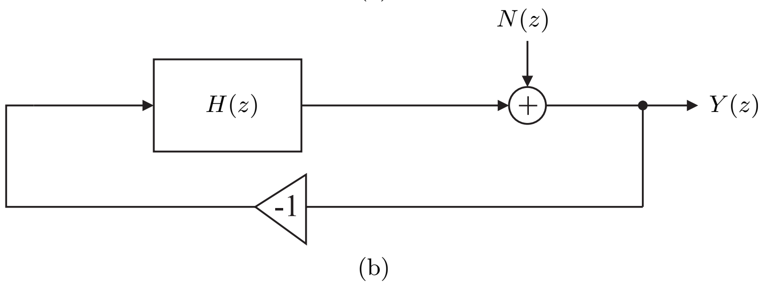


Fig. 2. Models for deriving the signal transfer function $H_x(z)$ (a) and the noise transfer function $H_n(z)$ (b).

This finally gives the $\Sigma\Delta$-converter depicted in Figure 3 with the transfer functions for the signal and the noise:

$$H_x(z) = \frac{\frac{z^{-1}}{1-z^{-1}}}{1+\frac{z^{-1}}{1-z^{-1}}} = z^{-1} \tag{11}$$

$$H_n(z) = \frac{1}{1+\frac{z^{-1}}{1-z^{-1}}} = 1 - z^{-1}\,. \tag{12}$$

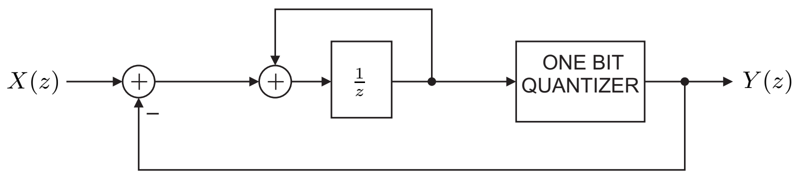


Fig. 3. $\Sigma\Delta$-converter with $H(z) = \frac{z^{-1}}{1-z^{-1}}$.

These transfer functions are depicted in Figure 4. As intended, the quantization noise is shifted to high frequencies, so that an analog lowpass at the output will suppress this noise and reconstruct the input signal.

Note, that the noise transfer function $H_n(z)$ is a first order highpass. Thus, the slope will be only 20 dB per decade. This can be improved furthermore by increasing the order of the $\Sigma\Delta$-converter as outlined in the next section.

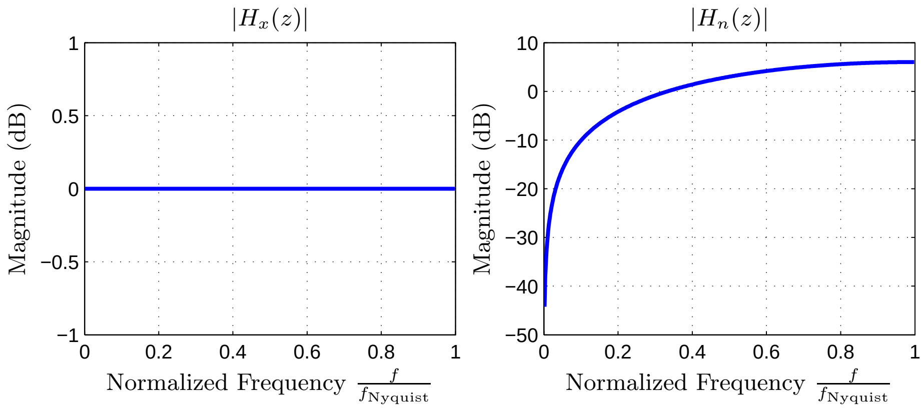


Fig. 4. Signal and noise transfer function of the $\Sigma\Delta$-converter depicted in Figure 3. As intended, the quantization noise is shifted to high frequencies.

3 Implementation of $\Sigma\Delta$-Converter

The $\Sigma\Delta$-converter can be used within a complete audio DAC as proposed in [1]. The structure of such a DAC is shown in Figure 5. It consists of the interpolator, the sigma-delta modulator, and the 1-bit DAC. The audio DAC accepts PCM input data at sampling rates of 32/44.1/48/88.2/96 kHz. The interpolation ratio of the interpolator can be configured to 64x, 128x, and 192x. For 44.1/88.2 kHz input signals, the interpolator gives the output data rate of 5.6448 MHz by setting the interpolation ratio as 128x/64x respectively. For 21/48/96 kHz input signals, the interpolator gives the output data rate of 6.144 MHz by setting the interpolation ratio as 192x/128x/64x respectively.

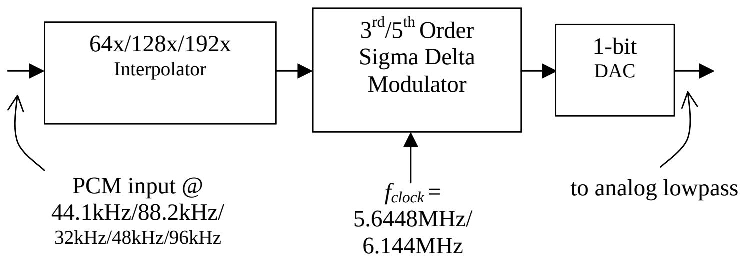


Fig. 5. Block diagram of the audio DAC.

The configurable $\Sigma\Delta$-converter proposed in this paper can be freely configured and can be used for the $\Sigma\Delta$-converter of the complete audio DAC. Our approach to create a soft core which is based on a VHDL description that can be configured to produce a $\Sigma\Delta$-converter of arbitrary order.

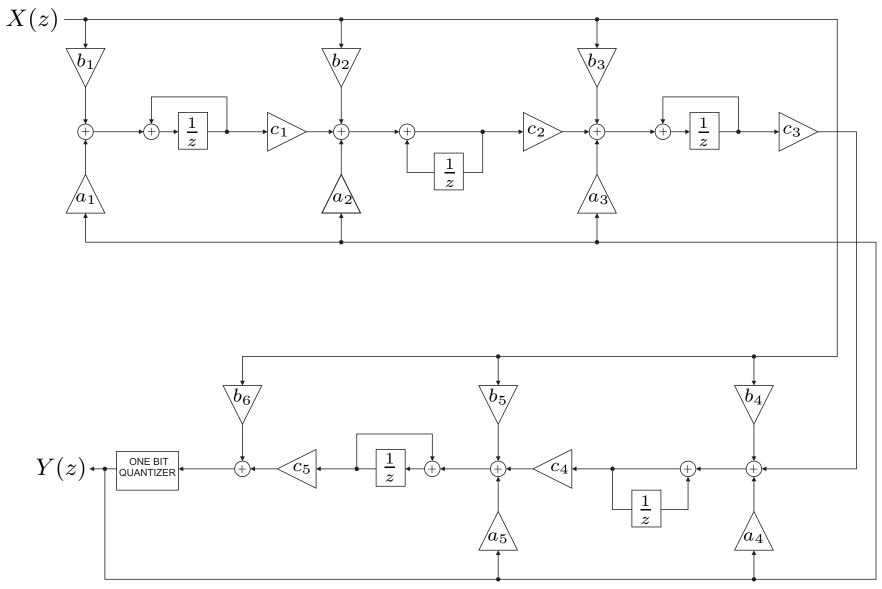


Fig. 6. $\Sigma\Delta$-converter architecture of the $\Sigma\Delta$-converterṪhe stages are generated using a `for-generate`-loop, so the number of stages can be adapted to the desired signal-to-noise-ratio.

As a starting point we used the architecture (see Figure 6) as proposed [1] for the $\Sigma\Delta$-converter which can be configured as either 3^{rd} order and as 5^{th} order modulator. The modulator coefficients were designed by using a Matlab toolbox [3].As it can be clearly seen the modulator is composed of two basic blocks (see Figure 7). Each of the two basic blocks is composed of a register for the delay and 3 multipliers. In VHDL, the constants a_n, b_n, c_n can be specified as a generic for every block separately. Everything is combined to the final architecture in the top-level design that uses a generate-loop to create a $\Sigma\Delta$-converter of the specified order.

While the system level simulation uses floating point numbers, a fixpoint implementation had to be derived for the FPGA realization. For a $\Sigma\Delta$-converter with a bitwidth of n at the input, the parameters of the stages have to be scaled by $2^{n-1} - 1$. Due to the integrating nature of the delay stages, it is not sufficient to use n as internal bitwidth. So the number of bits in the middle section of each stage (adder, delay and multiplication with c_n) must to be increased to avoid overflows.

As the VHDL description of the model is fully generic, a $\Sigma\Delta$-converter IP-core of the desired order can be generated very easily by changing the value

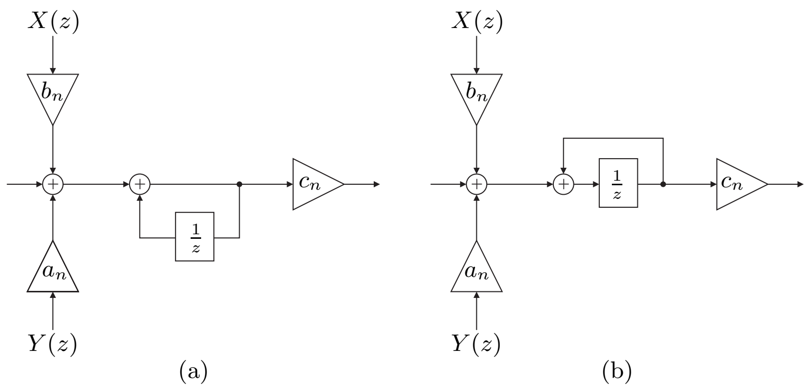


Fig. 7. Basic stages of $\Sigma\Delta$-converter

of the generics. The design parameters like the number of stages, the fixpoint bitwidth, the internal bitwidth and the converter coefficients are specified in a single configuration file.

4 System-Level Simulation

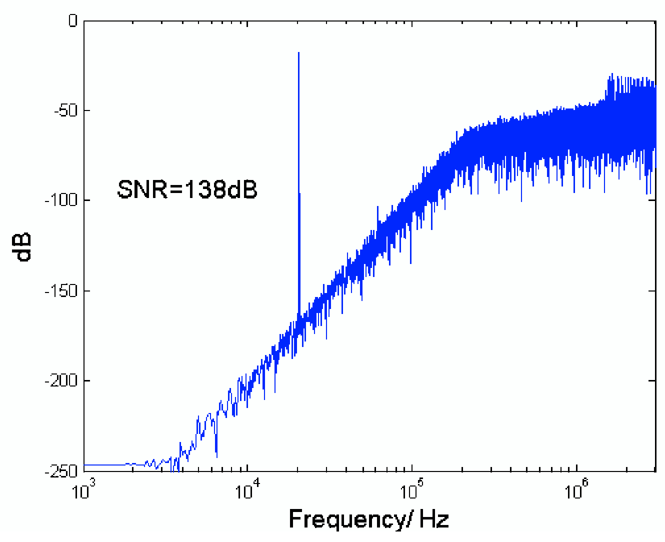
System level simulations were conducted using Matlab Simulink. Figure 8(a) shows the output spectrum for the 5th order configuration of the DAC, given a 20.48 kHz sinusoidal input signal (at the upper edge of the audio band), sampled at 96 kHz. It is then interpolated to a 6.144 MHz sampling rate using a two-stage sinc interpolator. The quantized 1-bit output of the sigma-delta DAC is then passed through an FFT to obtain the output frequency spectrum. The fifth-order noise shaping function is clearly observed. The audio-band SNR for this simulation is 138 dB. Figure 8(b) shows the output SNR versus the input signal level. From this figure, a maximum SNR of 140dB is obtained from the formula:

$$\text{Effective number of bits (ENOB)} = \frac{SNR - 1.76}{6.02},$$

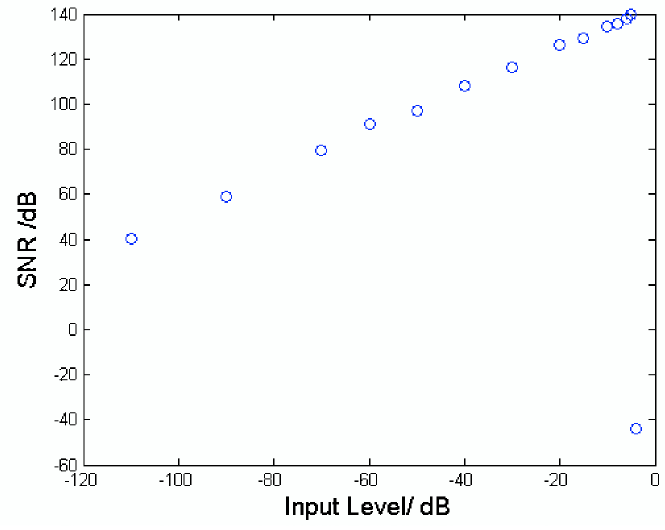
so this DAC configuration can operate on inputs with word lengths of up to 23 bits. The SNR performance versus input SNR for the 3rd order configuration of the DAC is shown in Figure 9. The maximum SNR is 96.4 dB, which corresponds to an ENOB of 15.7 bits. The system-level simulations verify the correctness of the DAC architecture.

5 Experimental Results

The complete design has been prototyped using an FPGA board with a XILINX XC2V1000-4 Virtex II FPGA. In order to produce some reasonable output, a 24-bit look up table containing one period of a 10 kHz sine wave sampled at



(a)



(b)

Fig. 8. System level simulation results for the 5th order configuration of the DAC. (a) output spectrum (b) output SNR versus input level.

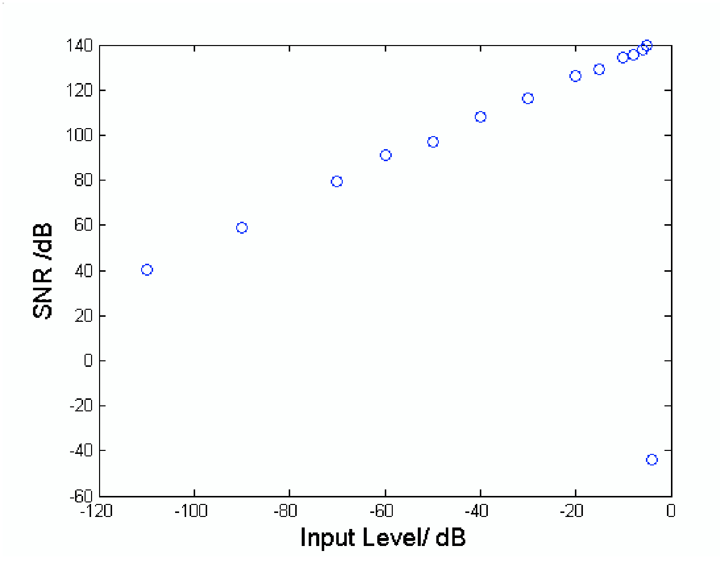


Fig. 9. Output SNR versus input level for the 3rd order configuration of the DAC.

6 MHz has also been mapped on the FPGA to provide a periodic input signal for the $\Sigma\Delta$-converter. In some designs (like in [1]) the c_n parameters of the $\Sigma\Delta$-converter are set to one so that the corresponding multipliers can be omitted. We have optimized our generic VHDL model so that the multipliers with a coefficient of one are automatically left out.

Table 1 shows the implementation details of our design. For a comparison with the results from the Handel-C implementation of [1] the first two values show the hardware resources and the maximum clock frequency for a coefficient set with $c_n = 1$. The other values are for a $\Sigma\Delta$-converter with all coefficients not equal to 1.

Table 1. Details of the implementation of a 5^{th} order $\Sigma\Delta$-converter with 24 bit input

Design	Slices	Multipliers	max. Clock
5^{th} order SD ($c_n = 1$)	362/5120	20/40	51 MHz
5^{th} order SD with LUT ($c_n = 1$)	967/5120	20/40	27 MHz
5^{th} order SD ($c_n \neq 1$)	566/5120	39/40	19 MHz
5^{th} order SD with LUT ($c_n \neq 1$)	1163/5120	35/40	16 MHz

It can be clearly seen that all the implementations meet the required 6 MHz operating frequency. Furthermore, using the $c_n = 1$ case for comparison with [1], this design uses the multipliers of the Virtex II device in order to achieve a large reduction in slice utilization (362 compared with 3167) as well as a higher operating frequency (51.7 MHz compared with 27 MHz). Even in the case of $c_n \neq 1$, only a modest number of slices are required. When synthesizing the design without the multipliers of the Virtex II device we also achive a lower slice utilization of 2757 compared to 3167 of the Handel-C approach (for $c_n \neq 1$).

To validate our implementation the spectrum of the output signal of the VHDL implementation was compared with the spectrum generated by the system level simulation. Furthermore we connected an analog low pass filter to the FPGA output and observed a very smooth and stable sine wave with a frequency of exactly 10 kHz.

6 Conclusions and Future Work

A $\Sigma\Delta$-converter-IP-core has been presented, that can be adapted easily to different requirements. This adaptation includes the selection of the bitwidth of the input signal and of the internal signals, the configuration of the coefficients and the $\Sigma\Delta$-converter core can be generated with an arbitrary number of stages. Additionally we included some automatic optimizations to remove unnecessary hardware (like a multiplication with one). With our VHDL $\Sigma\Delta$-converter-core we achieved much better results than with the Handel-C implementation.

The $\Sigma\Delta$-converter has been studied using simulink and both the system level simulation and the RT-level simulation results have been presented. The model has been prototyped on an FPGA board and the reconstructed sine wave generated by a LUT which has been also mapped onto the FPGA, could be observed using an simple analog low pass filter.

The next step is to replace the LUT with an interpolator in order to provide the required oversampling. Then the output of some conventional digital signal processing block (e.g. a MP3 decoder or a S/PDIF-receiver) can be used as input of the interpolator. Then it will be possible to listen to the output of the $\Sigma\Delta$-converter and judge the quality subjectively.

References

1. Ray C.C. Cheung and K.P. Pun and Steve C.L. Yuen and K.H. Tsoi and Philip H.W. Leong *An FPGA-based Re-configurable 24-bit 96kHz Sigma-Delta Audio DAC.* FPT 2003, Tokyo, Japan.
2. Gabor C. Temes and Shaofeng Shu and Richard Schreier *Architecture for Sigma Delta DACs*, in Delta Sigma Data Converters, edited by S.R. Norsworthy, et al, IEEE Press, 1997.
3. Richard Schreier, *http://www.mathworks.com*, MATLAB Central > File Exchange > Controls and Systems Modeling > Control Design > delsig.