University of Twente

System-on-chip

Continuous-time Sigma-Delta ADC

Final project group 4

Realization Report

Authors:
Daniël Huiskes s1937626
Pietro Pennestri s2382660
Giuseppe Recupero s2576414
Patrick Weske s1849379

S Gerez
R van der Zee

January 17, 2021

Abstract

This report discusses the realization of the analog and digital design of a sigma-delta modulator. An analog noise shaper is designed which shows a worst case scenario SNR of around 30dB. A 3 stage digital filter has been realized on a cyclone 5 `5CSEMA5F31C6` FPGA. The process includes a Simulink model, hardware design and a VHDL implementation. A hardware in the loop implementation has successfully been implemented to provide a realistic demo under the corona restrictions present during the whole project.

Contents

1 Introduction **2**

2 Problem overview **3**
2.1 Overall problem definition . . . 3
2.2 Design considerations . . . 3
2.3 Overall system design . . . 4

3 Analog realization **5**
3.1 Problem definition . . . 5
3.1.1 A first order sigma-delta modulator in Simulink . . . 6
3.1.2 SNR and Oversampling . . . 7
3.1.3 Second order sigma-delta modulator . . . 8
3.2 Design Considerations . . . 9
3.3 Realization . . . 11
3.3.1 Circuit functioning . . . 12
3.4 Measurement results . . . 14
3.4.1 SNR Analysis . . . 15
3.4.2 Analog Conclusion . . . 15

4 Digital realization **16**
4.1 Problem definition . . . 16
4.2 Design considerations . . . 16
4.2.1 Decimation . . . 16
4.2.2 Filtering . . . 17
4.3 Polyphase implementation . . . 18
4.4 Solutions . . . 19
4.4.1 Reference filter (Golden filter) . . . 19
4.4.2 Three stage filter . . . 20
4.5 Realization . . . 30
4.5.1 CIC . . . 30
4.5.2 FIR . . . 31
4.5.3 Complete filter . . . 34
4.5.4 Hardware in the loop . . . 36
4.5.5 Synthesis . . . 37
4.6 Measurement results . . . 44
4.6.1 CIC stage 1 measurements . . . 44
4.6.2 CIC stage 1 + FIR stage 2 + FIR stage 3 measurements . . . 45
4.7 Possible improvements . . . 45
4.8 Digital conclusion . . . 45

5 Delivered `vhdl` and script files **45**

1 Introduction

The goal of this project is to investigate, design and realize a delta sigma converter. The aim during this development process should be to create a modulator and a filter that operate according to the following specifications:

- At least 80 dB of signal to noise ratio (SNR)
- Analog input signal of $\approx 1V_{pp}$
- Digital ouput signal with a sample rate (f_s) of 48 kHz

The purpose of the ADC is to convert analog audio signals to digital signals. The frequency band for which this filter should be designed is, therefore, in the range of 20Hz up to 20kHz.

2 Problem overview

2.1 Overall problem definition

ADC (Analog to Digital Converter) is the bridge that links the analog world to the digital world. To preserve the quality of the signal, the converter must be accurate (high resolution, high SNR). It is difficult however to make an ADC with a very large resolution, as this would mean to increase N (number of bits) thus exponentially increase the number of components as well, (introducing mismatches and becoming easily impractical).

2.2 Design considerations

Oversampling sigma-delta modulation is one superior way to get a highly accurate converter. It takes the advantage of the oversampling technique and the noise shaping technique of the sigma-delta modulator, to perform a dual suppression of the quantization noise in the frequency band of the desired signal.

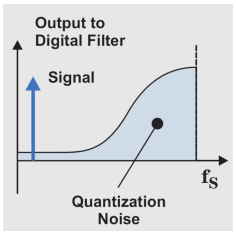


Figure 1: Signal and quantization noise in the spectrum, f_S is the oversampling frequency [1]

At the output of the modulator, we will already have our complete digital signal. However to fully convert it, we will need to filter the remaining quantization noise in the spectrum and downsample the signal to the wanted data rate, this will be performed by a digital filter and a decimator. Comparing to other converters, sigma-delta ADC has many advantages, such as high accuracy, high linearity and large dynamic range.

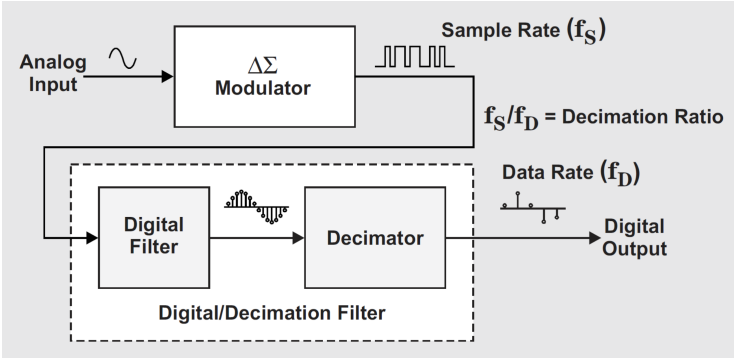


Figure 2: Block diagram of a sigma-delta ADC [1]

2.3 Overall system design

A block diagram schematic combining the analog and digital part is shown in Figure 3. This overview represents the essential functionalities of the analog and digital parts of the sigma delta converter design that were already discussed.

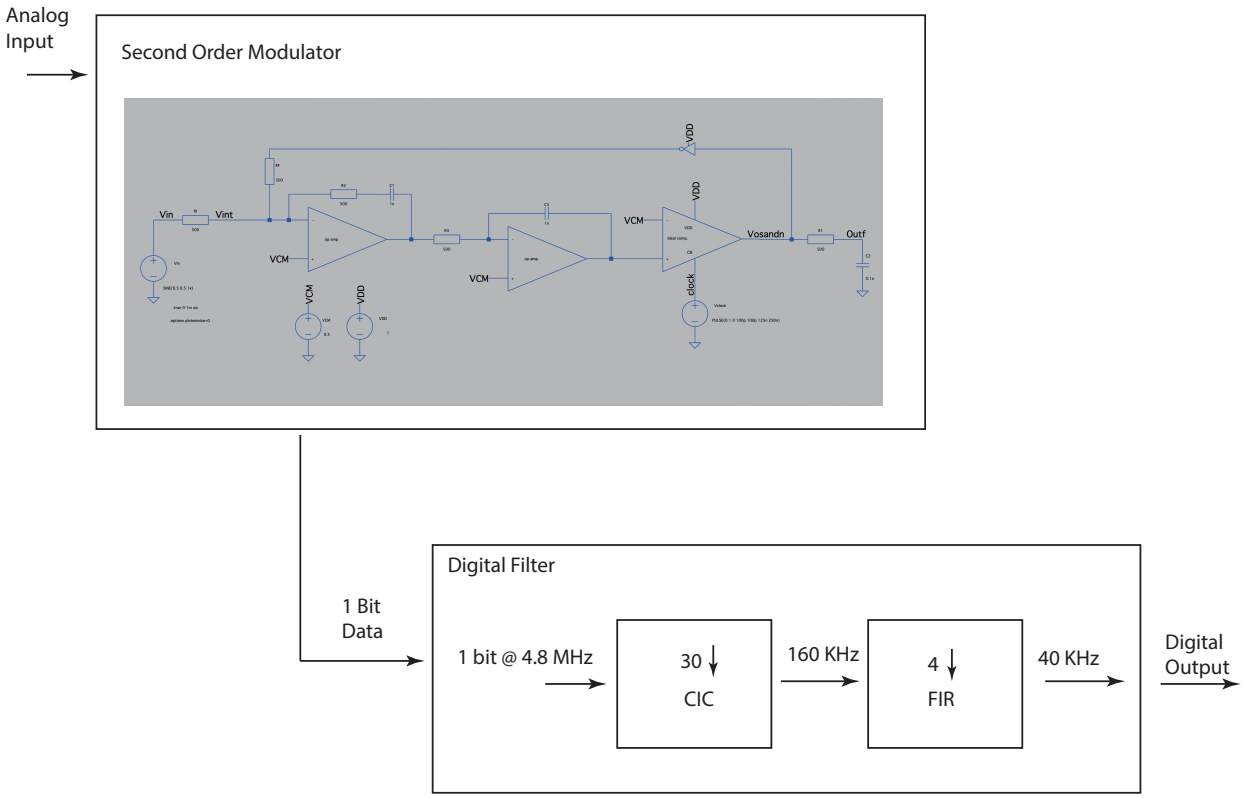


Figure 3: ADC block diagram

3 Analog realization

3.1 Problem definition

The core of a sigma-delta ADC is the analog modulator. The key point of the analog modulator is to act on the signal as a low pass filter and on the quantization noise as a high pass filter, shaping the noise out of the frequency band of the signal (this is why it is also called noise shaping modulator). This means that we can easily get a high SNR, especially for audio applications, where we will deal with a relative low frequency band of around 20 kHz.

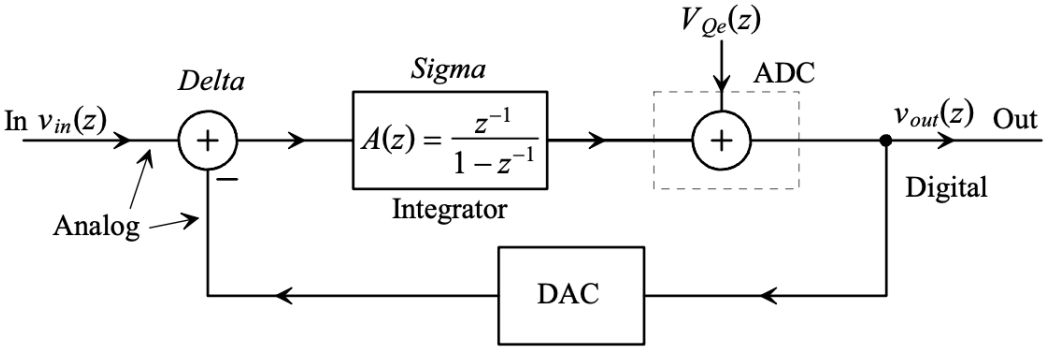


Figure 4: First Order Sigma-Delta modulator

The block diagram depicted in Fig. 4 represents the core of a sigma-delta modulator. The summer takes the difference (Delta) between the input signal and the fed back signal. The integrator accumulates or sums (Sigma) this difference and feeds the result back, via the ADC and DAC, to the summer. This forces the output of the modulator to track the average of the input. A sigma-delta modulator will be continuous time, when the sample and hold will be performed right after the integrator. In Fig. 4, the ADC can be seen as a zero order S/H and a comparator, making it a simple 1 bit ADC.

From this design, we can notice how the quantization noise is added at a different point compared to the signal, this is the key point for having two different transfer functions, one for the signal (STF) and one for the noise (NTF).

The complete transfer function for the output will be:

$$v_{\text{out}}(z) = \frac{A(z)}{1+A(z)} \cdot v_{in}(z) + \frac{1}{1+A(z)} \cdot V_{Qe}(z) \tag{1}$$

where

$$STF(z) = \frac{A(z)}{1+A(z)} = z^{-1} \tag{2}$$

$$NTF(z) = \frac{1}{1+A(z)} = 1 - z^{-1} \tag{3}$$

therefore, we can write the transfer function as:

$$v_{\text{out}}(z) = z^{-1} v_{in}(z) + \left(1 - z^{-1}\right) \cdot V_{Qe}(z) \tag{4}$$

We can see how in the z-domain STF is just a one sample delay, hence a low pass filter. While NTF is a differentiator, hence a high pass filter.

3.1.1 A first order sigma-delta modulator in Simulink

To verify the behavior of our sigma-delta modulator, a simulink analysis has been performed. The simulink model is depicted in Figure 5. The comparator is a simple sign operator, outputting 1 or -1 whether the signal is positive or negative, a zero order S/H will sample the output of the comparator at the clock frequency of 1 MHz. Both the input and output signal are then plotted as depicted in Fig.6

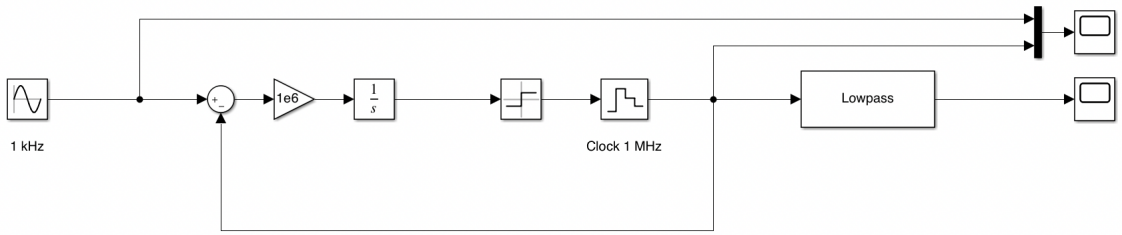


Figure 5: First order sigma-delta modulator, Simulink model

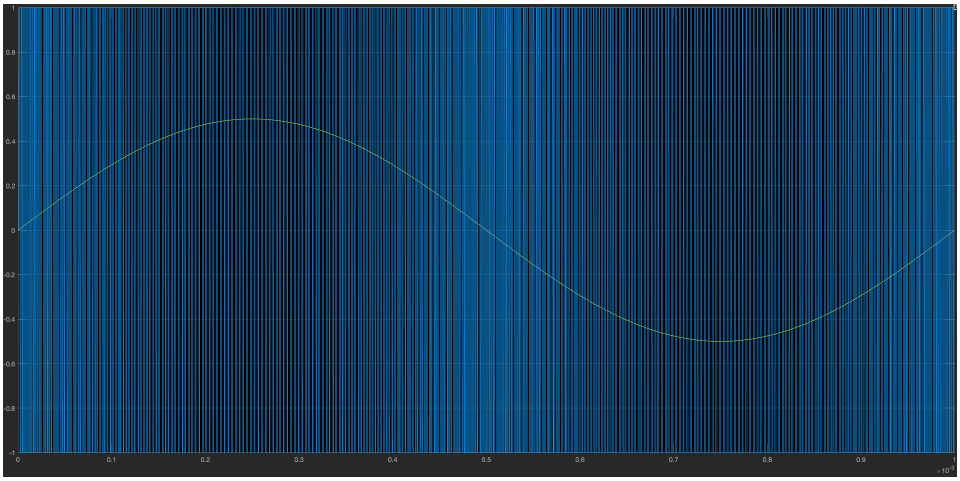


Figure 6: Analog input and digital output of the modulator

The modulator is performing correctly, since we can easily verify (Fig.7) that during the peak of the sine wave we will get more "1"s, at half period an equal amount of "-1"s and "1"s and for the negative peak more "-1"s. Therefore the digital signal really encodes the information of the input signal.

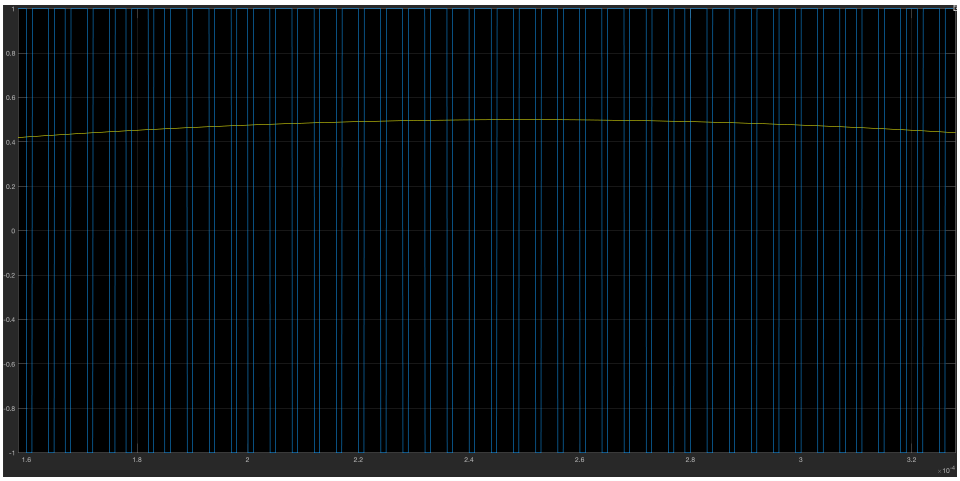


Figure 7: Detail of the peak of the sine wave

3.1.2 SNR and Oversampling

In "simple oversampling" according to the famous relation:

$$SNR_{ideal} = 6.02N + 1.76 + 10logK \tag{5}$$

and we can define, the increased resolution N_{Inc} as:

$$N_{Inc} = \frac{10logK}{6.02} \tag{6}$$

hence every doubling in the oversampling ratio, K, results in a 0.5-bit increase in resolution. In a sigma-delta modulator, however, the increased resolution will improve. Assuming a first order sigma-delta modulator, we will have [2]"Chapter 7, Eq.7.14":

$$SNR_{ideal} = 6.02N + 1.76 - 5.17 + 30logK \tag{7}$$

$$N_{Inc} = \frac{30logK - 5.17}{6.02} \tag{8}$$

therefore

$$SNR_{ideal} = 6.02(N + N_{Inc}) + 1.76 \tag{9}$$

hence every doubling in the oversampling ratio results in 1.5 bits increase in the resolution (9 dB increase in SNR_{ideal}). We achieve 1 extra bit of resolution compared to the "simple oversampling".

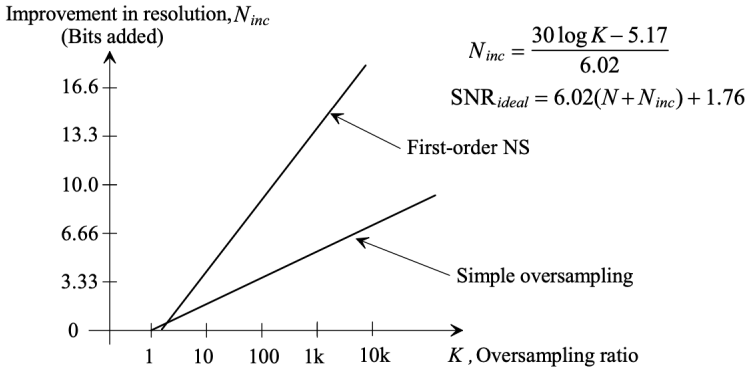


Figure 8: Improvement of a first order sigma-delta modulator[2]

3.1.3 Second order sigma-delta modulator

To improve the noise shaping of the modulator, we might perform additional quantization noise filtering. The second order modulator's output shows a double differentiation of the quantization noise, using two integrator stages.

$$v_{\text{out}}(z) = z^{-1}v_{in}(z) + \left(1 - z^{-1}\right)^2 V_{Qe}(z) \tag{10}$$

where

$$STF(z) = z^{-1} \tag{11}$$

$$NTF(z) = (1 - z^{-1})^2 \tag{12}$$

A **theoretical** implementation of the circuit is depicted in Fig.9

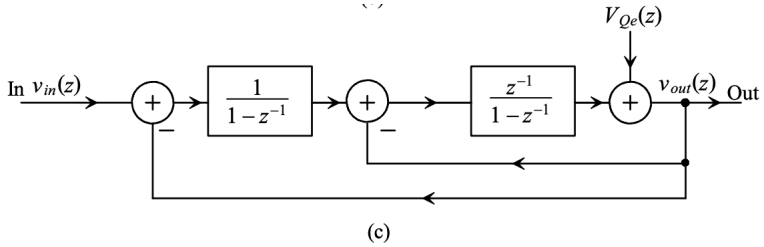


Figure 9: Second order modulator[2]

The modulation noise will be flatter in the bandwidth of interest as depicted in Fig.10

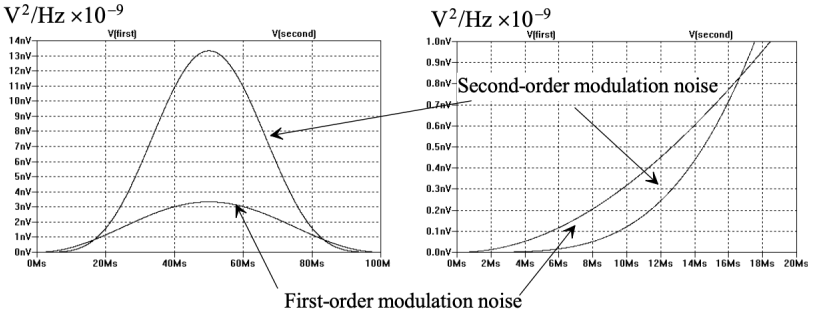


Figure 10: Comparing first and second order modulation noise[2]

with an increase in the SNR of

$$SNR_{ideal} = 6.02N + 1.76 - 12.9 + 50logK \tag{13}$$

Every doubling in the oversampling ratio results in an increase in SNR of 15 dB or 2.5 bits increase in resolution.

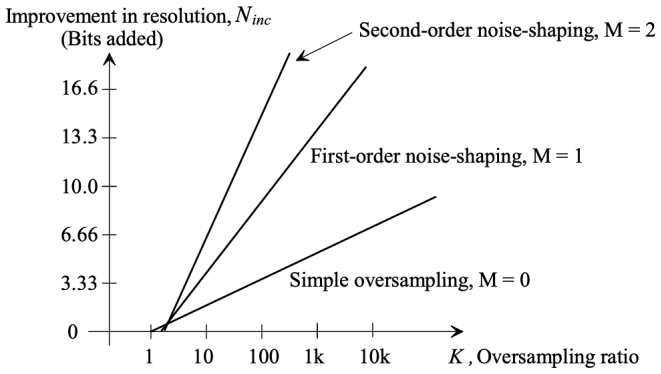


Figure 11: Improvement in modulator resolution[2]

Higher order modulators can be designed with a progressive increase in resolution and a better SNR. However, it will be substantially more difficult to ensure the stability of the system. A second-order topology (if correctly designed) will already perform better on the noise shaping than a first-order. An explicit evaluation regarding the SNR will be discussed in the following sections, and this led to the final choice of a second-order modulator.

3.2 Design Considerations

In a continuous time sigma-delta modulator the signal is not sampled at the input, but after the quantizer. The quantizer's output waveform is subtracted from the input and processed by a continuous time loop filter. Therefore the use of switched-capacitor integrators won't be required. In fact the design of such integrators becomes very challenging in low voltage CMOS technologies (it's difficult to to turn the switches on and off at low supply voltages), moreover a high sample rate (high clock rate) results in high power dissipation.

An actual implementation of a second-order modulator has been performed in LTSpice as depicted in Fig.12, using ideal components as a starting point. It consists of two integrators, one clocked comparator (quantizer) and a single feedback loop. The output signal after being inverted will be subtracted from the input. The reason for R2 is stability. Without the resistor R2, each filter can give a phaseshift of 90 degrees. Cascading two filters with 90 degree phaseshift can invert the signal, which in combinatiobn with a negative feedback loop causes a positive feedback loop. A positive feedbackloop is not stable if it has unity gain or higher. To circumvent this issue the resistor R2 is added, because at high frequencies, the capacitor is a short, but the R2 causes a resistive divider. A resistive divider has no phaseshift. Even if the second filter added it's maximum 90 degreees, the circuit would still be stable.

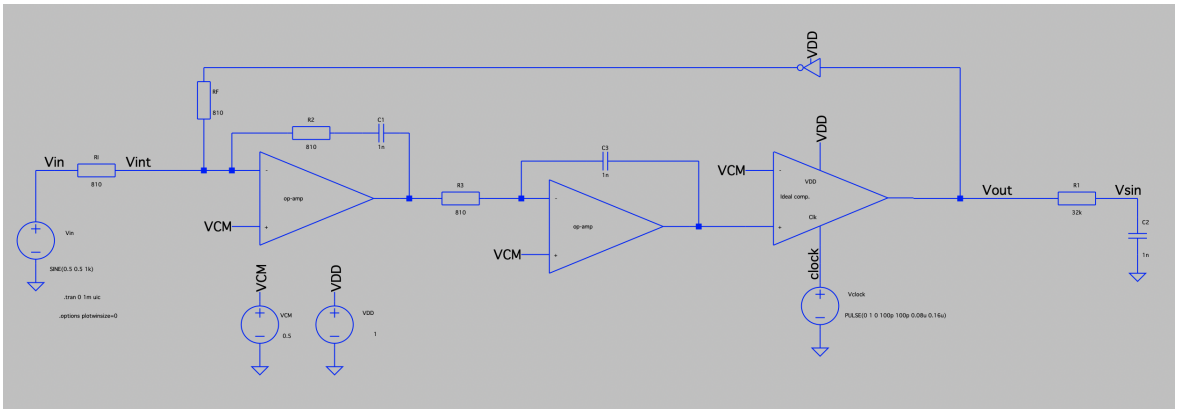


Figure 12: Ideal second-order modulator [2]

The value of RC must be much longer than the period of the sampling clock, T_s to keep v_{int} (the input voltage of the integrator) from varying too much. Setting, for instance, $RC = 5T_s$ we can compute the ideal SNR.[2]"Chapter 6, eq.6.77".

$$SNR_{ideal} = 6.02N + 1.76 - 40.85 + 50logK \tag{14}$$

therefore the bits increase will be

$$N_{inc} = \frac{50logK - 40.85}{6.02} \tag{15}$$

hence

$$SNR_{ideal} = 6.02(N + N_{inc}) + 1.76 \tag{16}$$

For the required $SNR = 80dB$, we can first calculate the N_{inc} from Eq.16 and then the oversampling ratio K from Eq.15. Given our Nyquist bandwidth of 48 KHz, **ideally** this will result in:

- $N_{inc} = 12$ bits
- K = 112
- $f_s = 5.37$ MHz

However we have to take into account the constraints coming from the digital design and the nature of the FPGA. Hence, the parameters for this design has been evaluated in Section 4.2. The value of RC = 800 ns has been chosen for the actual sample frequency of 6.25 MHz as depicted in Table1, "Chapter 4.2.2". The simulation results, using the real design parameters, are depicted in Fig.13, where we can observe the digital output of the modulator. Moreover, for the sake of the demonstration, a simple RC filter with a cut-off frequency above the input signal will show how the original signal can be reconstructed, Fig.14

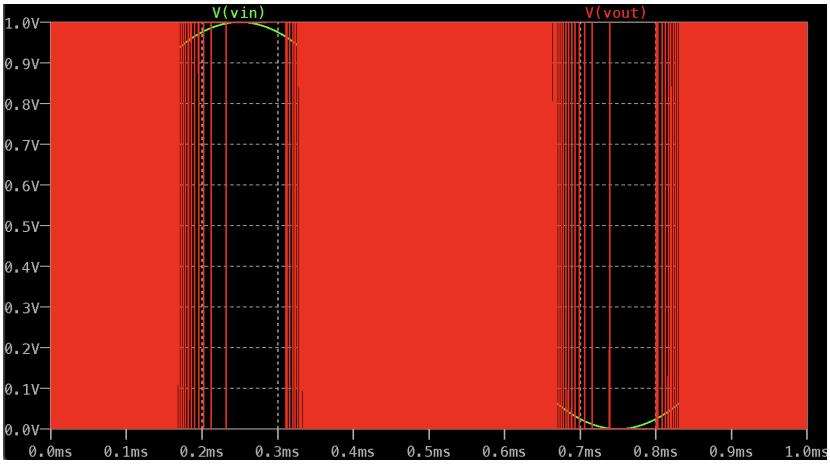


Figure 13: Ideal second-order modulator output

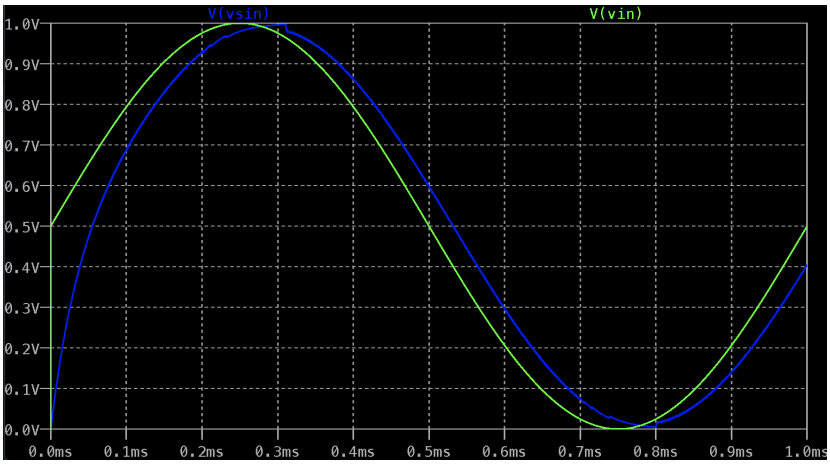


Figure 14: Input "Vin" vs Simple filtered output "Vsin"

3.3 Realization

The actual realization of the design has been implemented on a prototype board, using the following components:

- 1x AD8044
- 1x MAX942
- 1x SN74F74
- 820Ω, 10kΩ, 100kΩ Resistors
- 1nF, 4.7nF, $1\mu F$, $10\mu F$, $100\mu F$ Capacitors

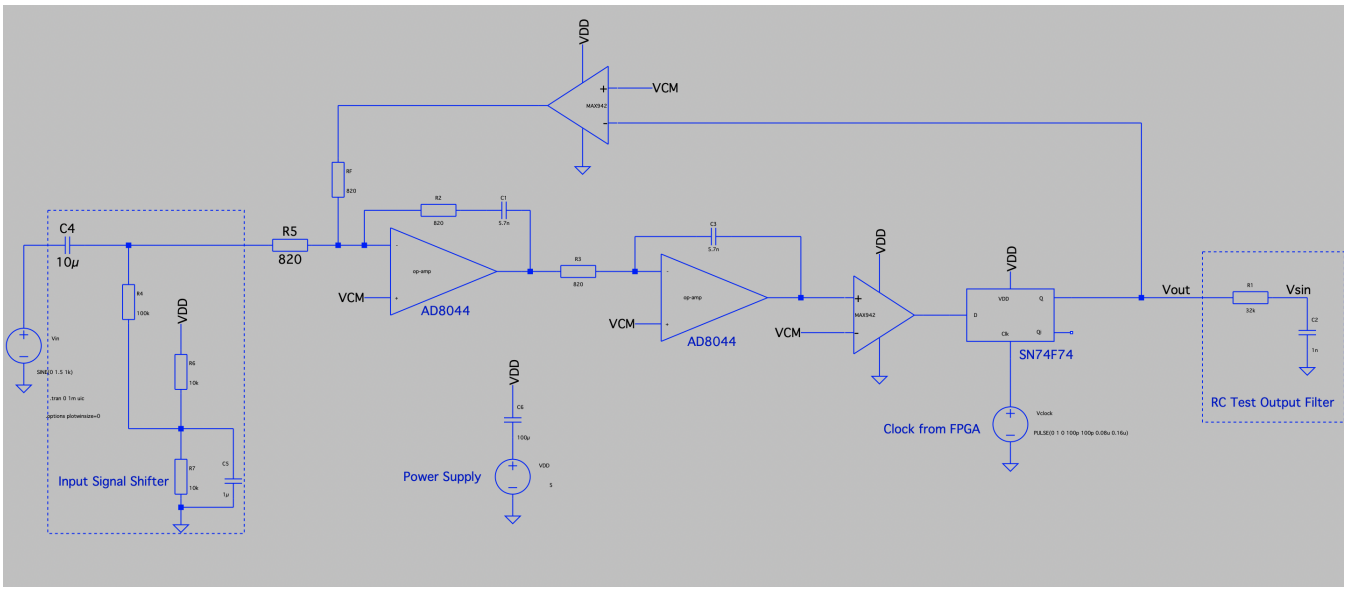


Figure 15: Schematic of the actual realization

3.3.1 Circuit functioning

Since the input signal will vary from negative to positive values and the circuit so far can only handle positive voltages, a "signal shifter" has been designed, as depicted in Fig.15. A resistor is placed between the input and the virtual ground to bias the input. Then a capacitor is connected between this node and the input to only let the AC be determined by the input and the DC offset by the bias. It should be noted that this capacitor and resistor form a high-pass filter, which has to be tuned to a cutoff frequency below 20 Hz in order to not attenuate the audio band. The virtual ground "VCM" is the middle between the resistors and will be used in the circuit as voltage reference for the opamps, as well for the comparators. The pole of the second integrator will be above the frequency band, set at around 34 kHz. Finally, to test the functionality of the modulator, the output will be filtered with a simple RC lowpass filter, showing that the information has been preserved.

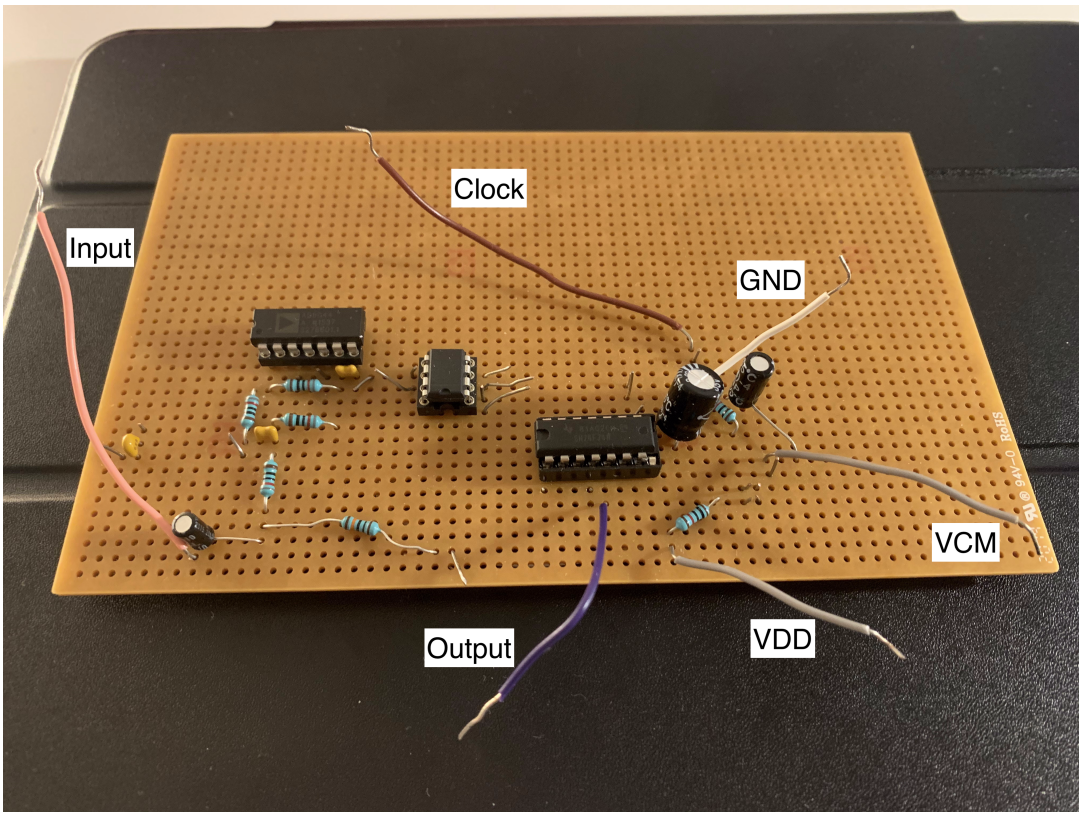


Figure 16: Prototype board design

3.4 Measurement results

Measurements and analysis of the results (even if it was not possible to merge the digital and analog part due to the separation between the 2 teams) has been performed in the lab. The equipment consists of 2 function generators (one for the clock, one for the input signal), 1 power supply and 1 oscilloscope.

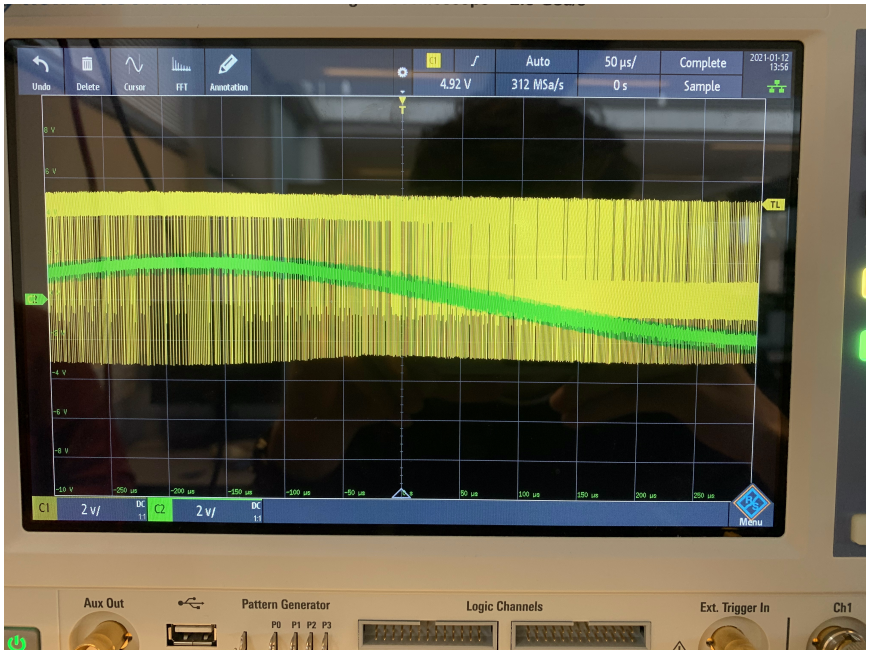


Figure 17: Input signal (green) and output digital signal (yellow)

The first measure shows how feeding an input sine wave to the modulator actually outputs a digital stream (square waves). Visibly showing more 1s at the top peak and more 0s at the bottom peak, as depicted in Fig.17. It is possible now to reconstruct the original sine wave using an RC filter with a cutoff above the frequency band, the measure is depicted in Fig.18. The output signal preserves the original information and it will be shifted at the reference voltage.

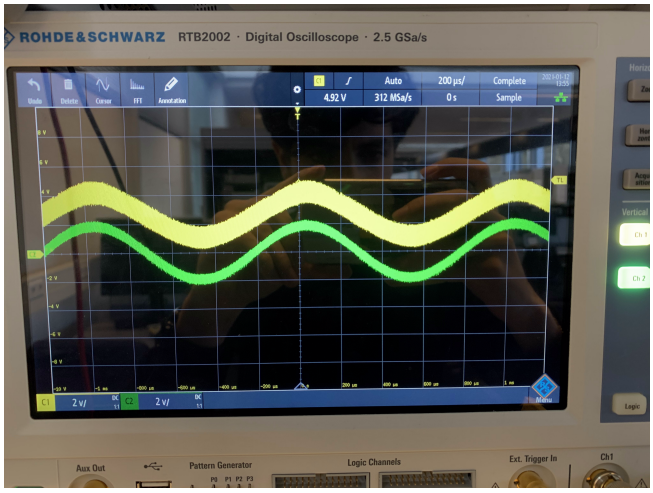


Figure 18: Input signal (green) and filtered output signal (yellow)

3.4.1 SNR Analysis

SNR analysis will evaluate the functioning of the analog modulator. Note that this measure has been performed without digital filtering involved. Therefore using the oscilloscope to perform an FFT on the output signal (filtered by an RC filter at 28 kHz) will allow to numerically calculate the SNR. The signal peak at -3dB (20kHz) and the noise floor level are highlighted in Fig.19.

Figure 19: FFT of the output signal

Proceeding with the calculation, we have to detect the noise power in the bandwidth, in our case 50kHz.

$$NoisePower = -70dB + 10log(50kHz/10.65Hz) = -33.28dB \tag{17}$$

Hence,

$$SNR = -3dB + 33.28dB = 30.28dB \tag{18}$$

3.4.2 Analog Conclusion

In conclusion, the worst case scenario shows an SNR of around 30dB. However to obtain a better estimate of the overall SNR, the measure should have been performed after digital filtering and using a real tone/audio signal interfacing with a computer. In fact, the input signal from the function generator was already quite noisy. It appeared that the clock generator, without interfacing with the circuit, when turned on would add harmonics to the input generator, making it substantially more noisy.

4 Digital realization

4.1 Problem definition

The analog part of the sigma delta ADC has to be linked to the digital part of the ADC. Important considerations are that the input signal of the digital part will have a certain oversampling rate and besides that noise is shaped to higher frequencies.
Therefore a digital filter should be designed that in essence should perform two tasks [3].

- Removing the noise that is shaped to the higher frequencies
- Apply Decimation to the analog input signal to downsample it to the desired 48 kHz f_s

By removing the high frequency noise that is outside of the baseband only a small amount of noise remains. By doing this the resolution of the signal will effectively be increased [3].
Besides that the principle of decimation is essential to reduce the signal back to the Nyquist rate, which minimizes the amount of information that can be used for further processing[3].

In the design of this filter structure it is important to take the influence of the filter to the SNR into account, such that the output signal of the digital part preserves a SNR around 80 dB.

4.2 Design considerations

4.2.1 Decimation

Decimation is the process of down sampling the signal to the desired rate. In the process of decimation often a lowpass filter is used. The decimation is done by by only keeping the M^{th} samples from the signal. By doing this the sample rate will effectively be reduced by a factor of M.
A filter that conveniently can be used for this purpose is a sinc filter. Sinc filters have a sharp filter characteristic in the frequency domain. However sinc filters can not directly be implemented, but have to be implemented using FIR filter structures. These have a less ideal filter characteristic, but by carefully choosing the design and filter order it is possible to achieve the desired filter characteristics.
This filter has 2 tasks. It should achieve the desired down-sampling ration and filter out the high frequency quantization noise.
This can be accomplished by cascading a simple FIR filter with a more complex FIR filter [4]. An example of a general FIR filter structure is shown in figure 20 and an example of a cascaded structure is given in figure 21.

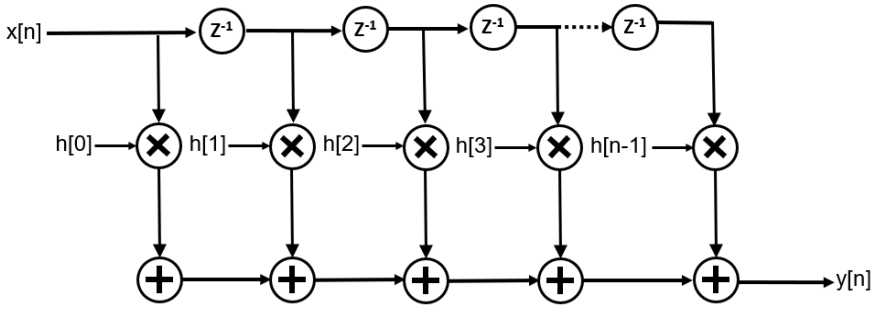


Figure 20: Structure general FIR filter

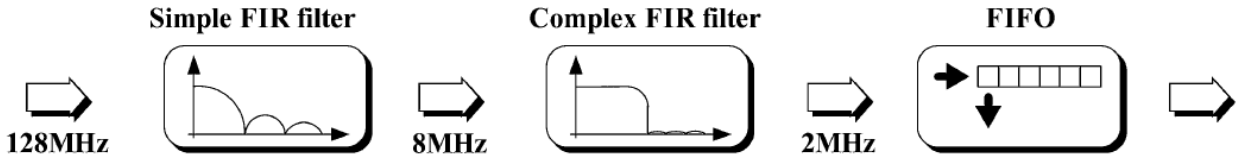


Figure 21: Multiple stages FIR filter [4]

The FPGA on which the digital system is implemented has a clock frequency of 50MHz. This implies that the frequencies that the sampling frequencies that can be used with this clock have to follow equation 19. Therefore f_s of the input signal is 6.25MHz. This sample frequency was also mentioned and explained in the analog part of this report in subsection 3.2. The digital output should however be sampled with a frequency as close as possible to the nyquist frequency. Using equation 19 it can be found that this requires a decimation of n = 128, which results in a f_s of the output signal of 48.8kHz

$$f_s = \frac{50}{2^n} \text{ MHz} \tag{19}$$

4.2.2 Filtering

In this example shown in figure 22 first a simple FIR filter is used to decrease the sampling frequency with a factor of 16. Subsequently a more complicate FIR filter is used to further decrease the sampling frequency with a factor of 4. The advantage of using a structure like this is that first a decimation is made with a simple filter that works at a high frequency and that a smaller decimation is made with the slower filter stage. The cut-off frequency of the simple filter is not as constraint as the cut-off frequency of the complex filter. The main part of the filtering of the quantization noise is done in the first stage.
The simple FIR filter can be implemented in the form of a so called cascaded CIC filter. This is a simple filter that consists out of integrator cascaded with combs. It is therefore not exactly a FIR filter, but has some similarities. An example of such a CIC filter structure is shown in figure 22. This figure shows a CIC filter that makes use of 3 stages. The I's in the diagram represent the integrators and the C's the comb structures. The R in between them indicated the reduce of sampling rate between them. This is essential for decimation. The transfer function for the CIC filter is shown in equation 20. In this equation R is the rate change, N is the number of cascaded stages and M is the differential delay which is often set to 1[5].

$$H(z) = \frac{(1 - z^{-RM})^N}{(z^{-1})^N} = \left(\sum_{k=0}^{RM-1} z^{-k} \right)^N \tag{20}$$

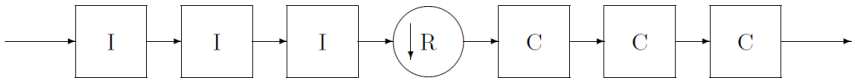


Figure 22: CIC filter - three stages [5]

The relation in equation 20 allows for a separate integration and differentiation for the filter.
The second filter stage can be used to remove the quantization noise from the signal and further reduce the sample rate.

Important characteristics for the filter are the upper edge of the passband f_p, which can be calculated using equation 21 and the lower edge of the stopband which can be calculated with equation 22, where OSR is the oversampling ratio [6]. These equations can be used to determine the minimum alias protection that

is given when a first-order alias filter is used. The equation that provides this relation is given in equation 23 [6]. The required filter order can subsequently be determined using equation 24. The calculations that follow from these equations give enough information to design a dedicated FIR filter using Matlab filterDesigner.

$$f_p = \frac{1}{2OSR} \tag{21}$$

$$f_i = 1 - \frac{1}{2OSR} \tag{22}$$

$$A_1 = \frac{\left|H_1(e^{j2\pi f_p/N})\right|}{\left|H_1(e^{j2\pi f_i/N})\right|} = \left|\frac{sin(\pi f_p)sin(\pi f_i/N)}{sin(\pi f_p/N)sin(\pi f_i)}\right| \tag{23}$$

$$\text{required sinc filter order} = \frac{100}{20log_{10}(A_1)} \tag{24}$$

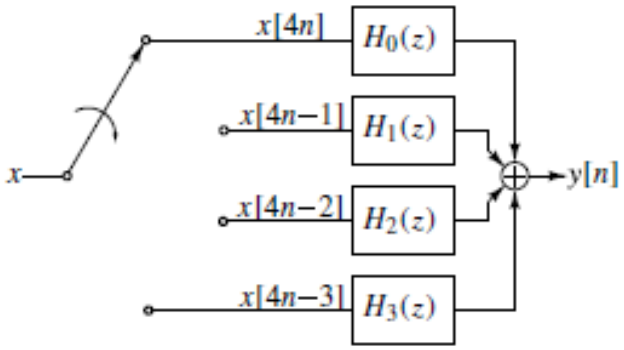
A summary of the system specifications is reported in Table 1.

Parameter	Symbol	Value
Approximate Signal Bandwidth	BW	24 kHz
Sampling Frequency	f_s	6.25 MHz
Over Sampling Ratio	k	128
Modulator Order	M	2
Output bits of the modulator	B	1
Increase in Resolution	N_{inc}	12
Minimum Signal to Noise Ratio	SNR_{min}	80 dB

Table 1: System Specifications

4.3 Polyphase implementation

Because decimation requires only taking every fourth output of the FIR filter it makes no sense to compute all of the values in between. Therefore it is possible to use a special structure that only takes the relevant samples into account. This increases the efficiency the the method by reducing the amount of calculations that have to take place [7]. An implementation that can be used for this is the polyphase implementation. A figure that illustrates this implementation can be seen in figure 23.



$$H(z) = H_0(z^4) + z^{-1}H_1(z^4) + z^{-2}H_2(z^4) + z^{-3}H_3(z^4)$$

Figure 23: Polyphase filter - order N = 4 [6]

4.4 Solutions

As already mentioned the Matlab filterDesigner tool will be used for the design of the decimation filter. This filter will be exported to the Matlab Simulink environment. By doing this the output of the analog model can be used as input for the digital model. The simulation results can be used to determine the signal to noise ratio. In order to find a suitable solution a trade off has to be made between the filter order of the used filter system and the quality of the filter output. This quality can be expressed in the SNR values and is very dependent on the stopband frequency, passband attenuation and stopband attenuation.

4.4.1 Reference filter (Golden filter)

In order to have a reference for more complex filter structures it was decided to first create a reference FIR filter. We refer to this filter as the golden filter, because it is an ideal FIR filter with a really sharp filter characteristic together with a lot of passband and stopband attenuation.

However, this filter is not suitable for implementation. This is shown by the fact that the filter order that comes out of the filter designer tool is much too high. The order is 2114, which means that this filter requires an enormous amount of hardware. Much more hardware than is needed to get results close to the result of this ideal filter. The behaviour of the designed reference filter is shown in figure 24. The hardware overview of the filter is shown in figure 25.

The sharp filter characteristic is is shown in the plotted filter behaviour by the sharp drop between the passband and stopband frequencies and the very small attenuation behind the stopband.

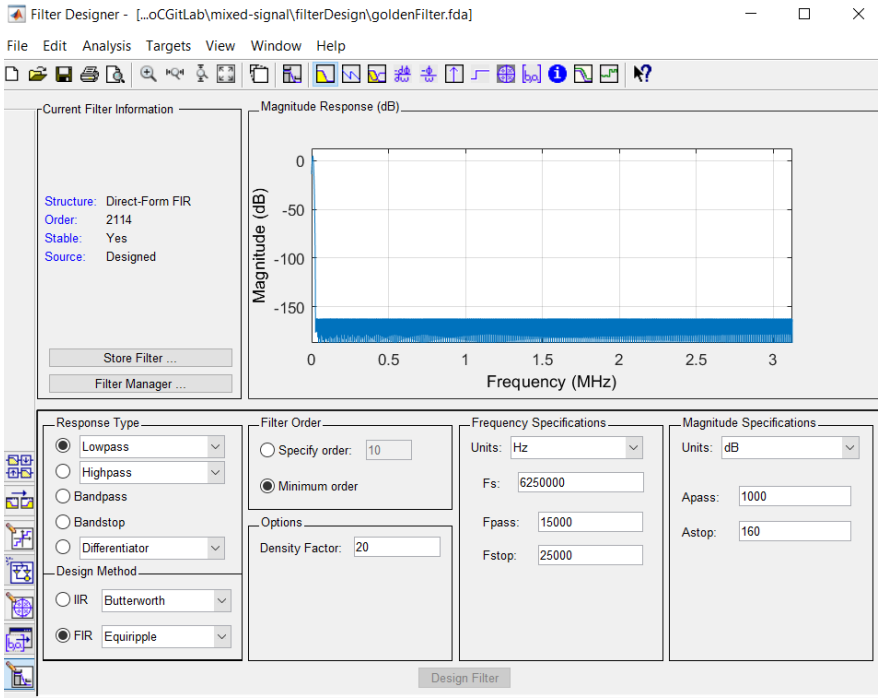


Figure 24: FIR reference filter

```
Reference - Discrete-Time FIR Filter (real)
-------------------------------
Filter Structure  : Direct-Form FIR
Filter Length     : 2115
Stable            : Yes
Linear Phase      : Yes (Type 1)

Implementation Cost
Number of Multipliers            : 2115
Number of Adders                 : 2114
Number of States                 : 2114
Multiplications per Input Sample : 2115
Additions per Input Sample       : 2114
```

Figure 25: FIR reference - information structure and hardware

4.4.2 Three stage filter

As already mentioned in the design considerations it is possible to use simple filters followed by a more complex filter that is specifically designed to filter out all higher frequency noise. This is also the approach that will be followed for the final solution. The approach is to use a CIC filter, which filters a little amount of noise, but which does most of the decimation process. This is followed by two FIR filter stages. Both stages perform a minor decimation. The first stage is a less sharp filter and the last stage is a sharper filter that should filter out most of the remaining noise. The total decimation factor of the filter has to be 128. This has been determined by using equation 14. As a decimation strategy there has been chosen to let the CIC filter perform a decimation of 32. Subsequently both FIR stages will perform a decimation of 2. The input signal has a wordlength of 1 bit and will be zero padded to a wordlength of 16 bits, which is used in the entire filter structure. A simple overview of this filter can be seen in figure 26. A similar approach was used by Kahlid [8]. In his work also a 3 stage filter was implemented, however there has been chosen to implement a different decimation strategy.

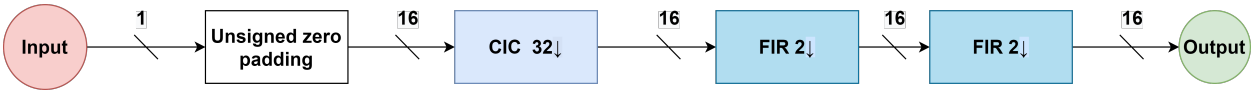


Figure 26: Overview of 3 stage filter

4.4.2.1 CIC stage

The CIC filter has a decimation factor of 32. The sample frequency of the input signal is 6.25Mhz. The sample frequency will therefore be 195.31kHz. There was chosen to implement this filter using 2 sections. The filter response is shown in figure 22. As can be expected from a CIC filter there is a little amount of attenuation over the higher frequencies. The structure is similar as in figure 22, only now there are 2 sections instead of 3. The hardware that is needed for CIC stage 1 is shown in figure 28.

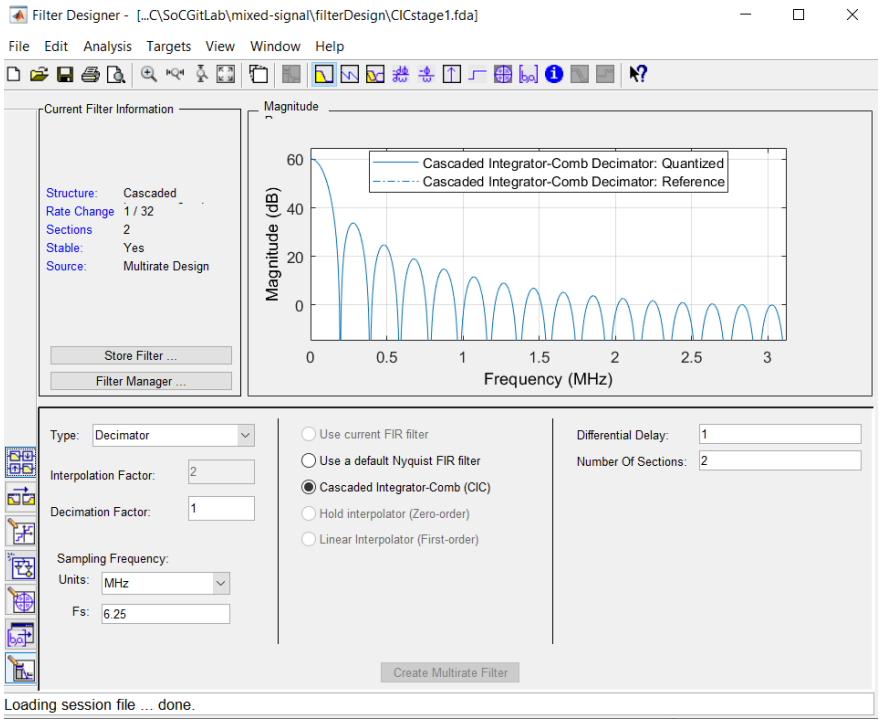


Figure 27: CIC filter design Simulink

```
CIC stage 1 - Discrete-Time FIR Filter (real)
-------------------------------
Filter Structure   : Cascaded Integrator-Comb Decimator
Decimation Factor  : 32
Differential Delay : 1
Number of Sections : 2
Stable             : Yes
Linear Phase       : Yes (Type 1)

Implementation Cost
Number of Multipliers            : 0
Number of Adders                 : 4
Number of States                 : 4
Multiplications per Input Sample : 0
Additions per Input Sample       : 2.0625
```

Figure 28: CIC stage 1 - information structure and hardware

4.4.2.2 Stage 2 FIR

The sample frequency of the input signal is 195.31kHz. The FIR filter has a decimation factor of 2 and therefore the output signal will be sample with a frequency of 97.66 kHz. The designed filter is shown in figure 29. It can be seen that there is chosen for a stopband frequency of 83kHz, a passband attenuation of 0.1dB and a stopband frequency of 100.

The values for the design parameters of the filter where found by trial and error. First an educated guess was done on which filter values could have beneficial filter characteristics while keeping the order as low as possible. Because most of the noise is filtered out in the last stage it was possible to be lenient on the stopband criteria. There was however chosen to have a tight passband attenuation to keep a clear signal and a larger stopband attenuation to minimize the influence of the noise in the higher frequencies. This is also reflected in the plot of the filter response. The filter is not sharp in the sense it has a slow decent and only gets sharper at the higher frequencies. However there is very little attenuation in the passband and also only a little amount of ripple in the stopband. The filter has an order of 8. The hardware overview is shown in figure 30.

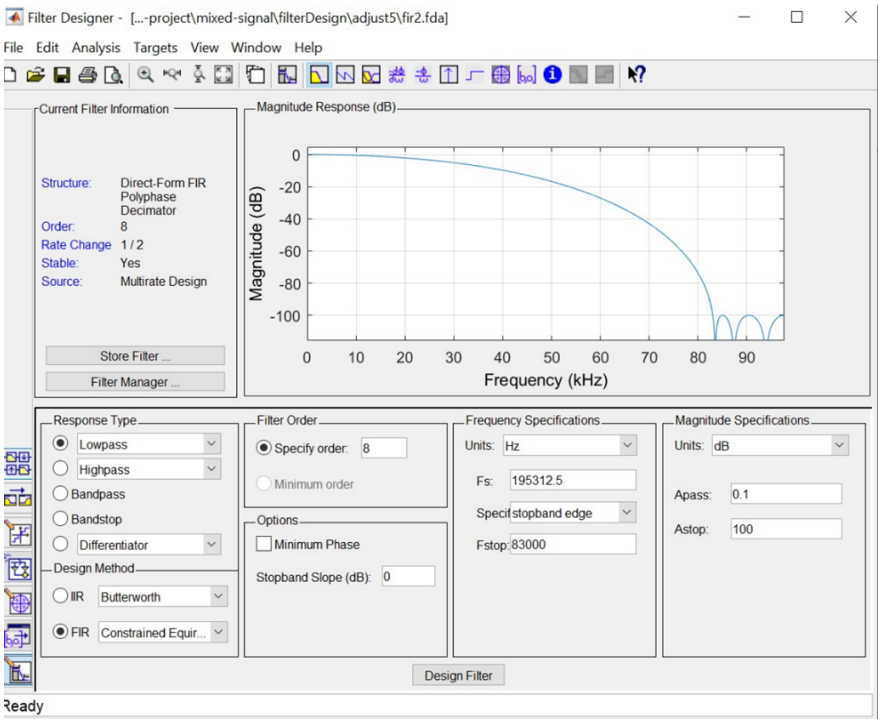


Figure 29: FIR filter stage 2

```
FIR stage 2 - Discrete-Time FIR Filter (real)
-------------------------------
Filter Structure   : Direct-Form FIR Polyphase Decimator
Decimation Factor  : 2
Polyphase Length   : 5
Filter Length      : 9
Stable             : Yes
Linear Phase       : Yes (Type 1)

Implementation Cost
Number of Multipliers            : 9
Number of Adders                 : 8
Number of States                 : 8
Multiplications per Input Sample : 4.5
Additions per Input Sample       : 4
```

Figure 30: FIR stage 2 - information structure and hardware

The filter as shown in figure 29 was realized in Simulink by using block mimicking components that are available in an FPGA to get a good representation of the filter that will be implemented in the FPGA. This realization can be seen in figure 31. The realization exists of two parts. One part that consists of a decimation stage and one filter atoms stage. An atom is the combination of a gain and a summation unit. This is also shown by a gray rectangle in figure 31. The structure in the decimation stage block can be seen in figure 32. The decimation in this stage is accomplished by 2 down samplers. In this structure there has also been made use of the polyphase principle. This structure is comparable to the structure as shown in figure 23, only in this implementation the output y[n] is the sum of 2 parts instead of 4.

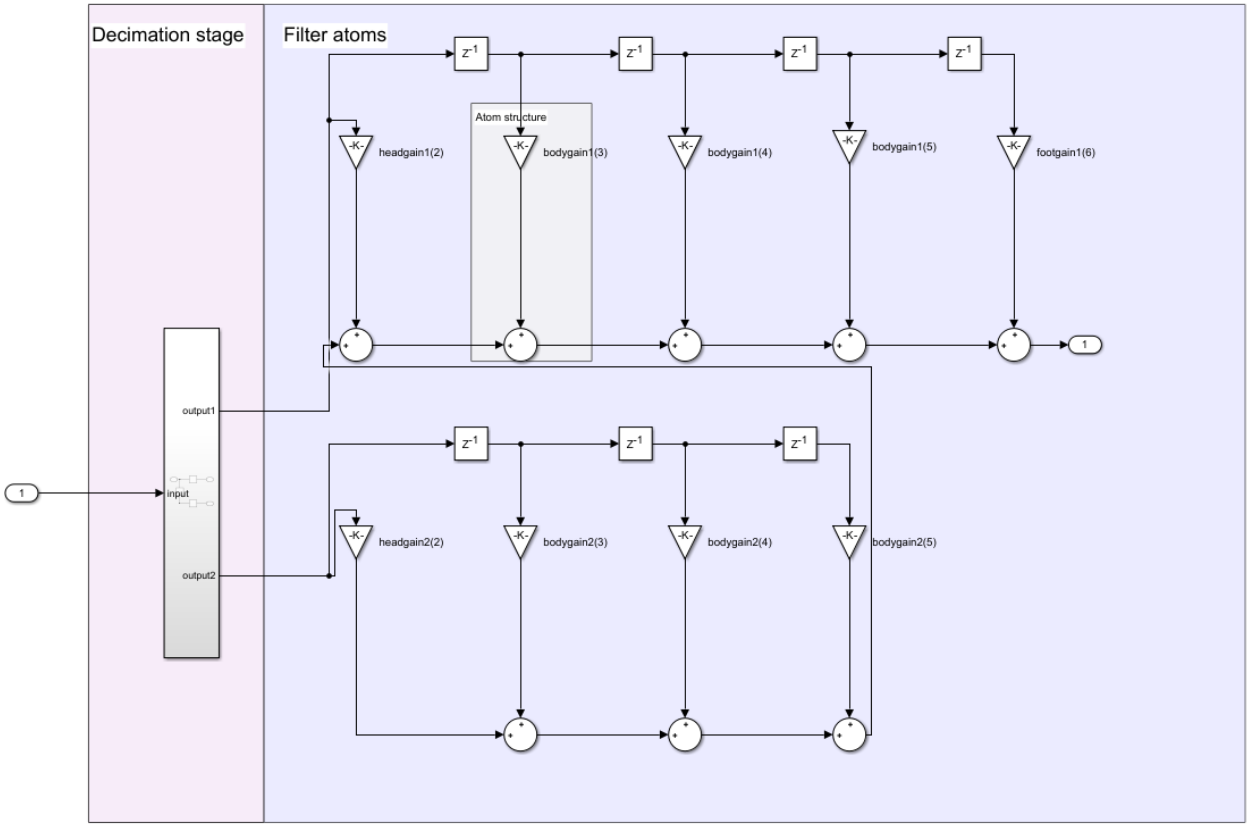


Figure 31: FIR stage 2 Simulink block model

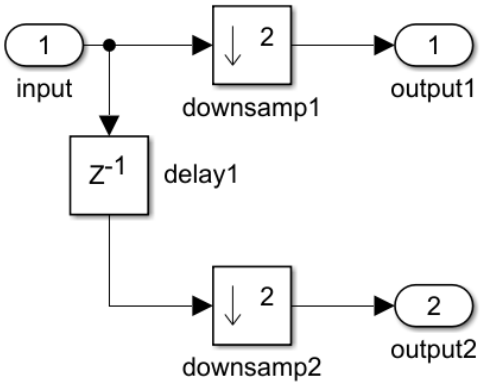


Figure 32: FIR decimation stage

4.4.2.3 Stage 3 FIR

Filter stage 3 is created to filter out most of the noise. The filter applies a decimation of 2. The input signal is sampled with a frequency of 97.66kHz and the output signal has a sample frequency of 48.83 kHz, very close to the Nyquist frequency. The designed filter response is shown in figure 33. The filer has a stopband frequency of 32kHz, a passband attenuation of 0.1dB and a stopband attenuation of 90dB. When looking at

the filter response it can be seen that this filter is sharper than the filter in the second stage. In order not to make the filter order too high, the stopband attenuation has therefore been slightly lowered compared with FIR stage 2. The order of this filter is therefore 18. Just as with FIR filter stage 2, a block model was made in simulink for FIR stage 3 that is shown in figure 35. The decimation stage is similar as in figure 32. Also the structure of the atoms is similar, only there are now more filter atoms due to the larger filter order. The hardware overview needed for this filter stage is shown in figure 34.

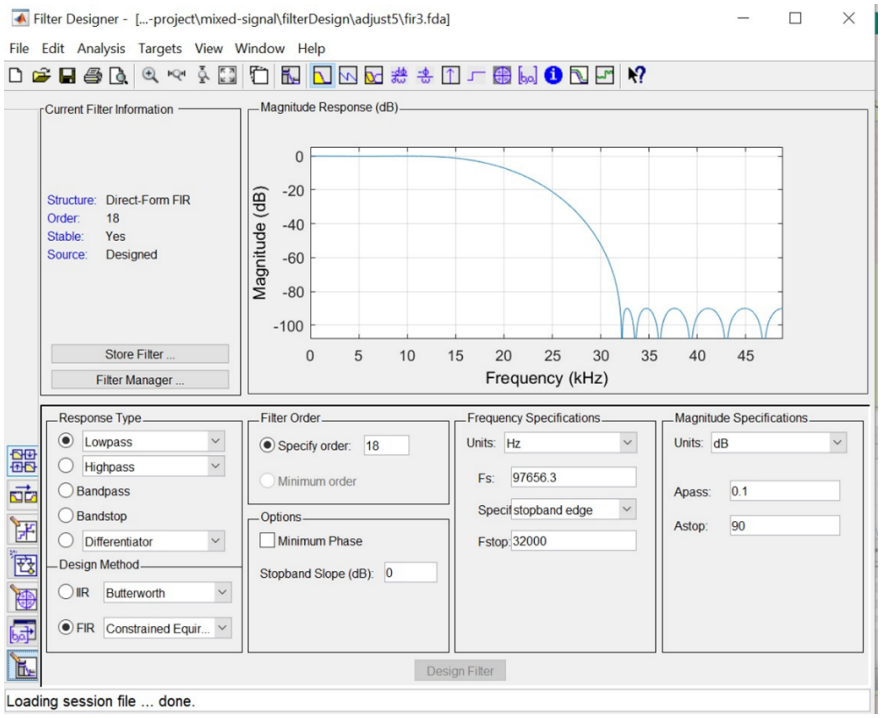


Figure 33: FIR filter stage 3

```
FIR stage 3 - Discrete-Time FIR Filter (real)
-------------------------------
Filter Structure   : Direct-Form FIR Polyphase Decimator
Decimation Factor  : 2
Polyphase Length   : 10
Filter Length      : 19
Stable             : Yes
Linear Phase       : Yes (Type 1)

Implementation Cost
Number of Multipliers            : 19
Number of Adders                 : 18
Number of States                 : 18
Multiplications per Input Sample : 9.5
Additions per Input Sample       : 9
```

Figure 34: FIR stage 3 - information structure and hardware

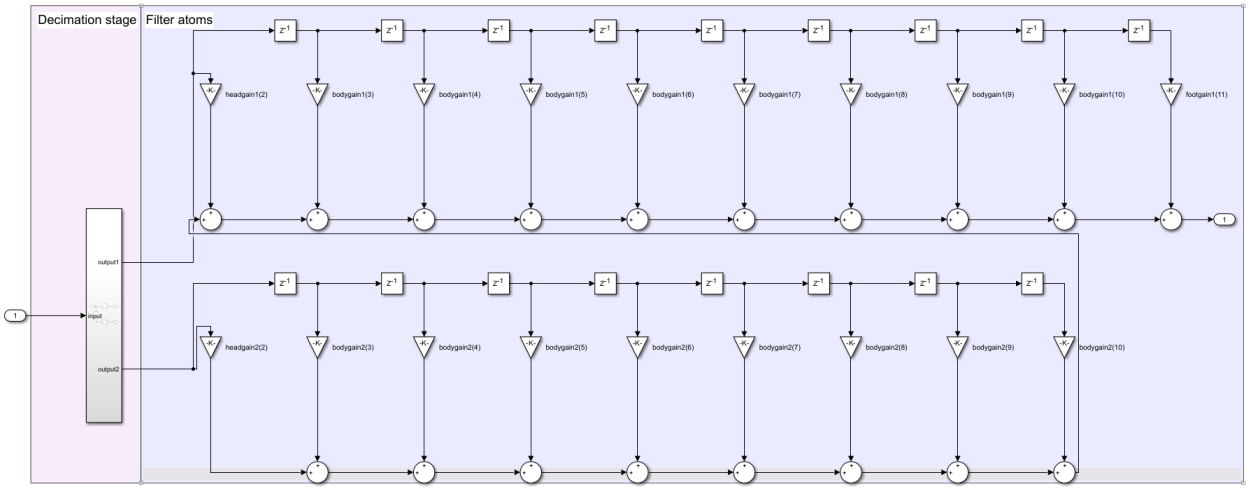


Figure 35: FIR stage 3 - simulink block model

4.4.2.4 Filter cascade

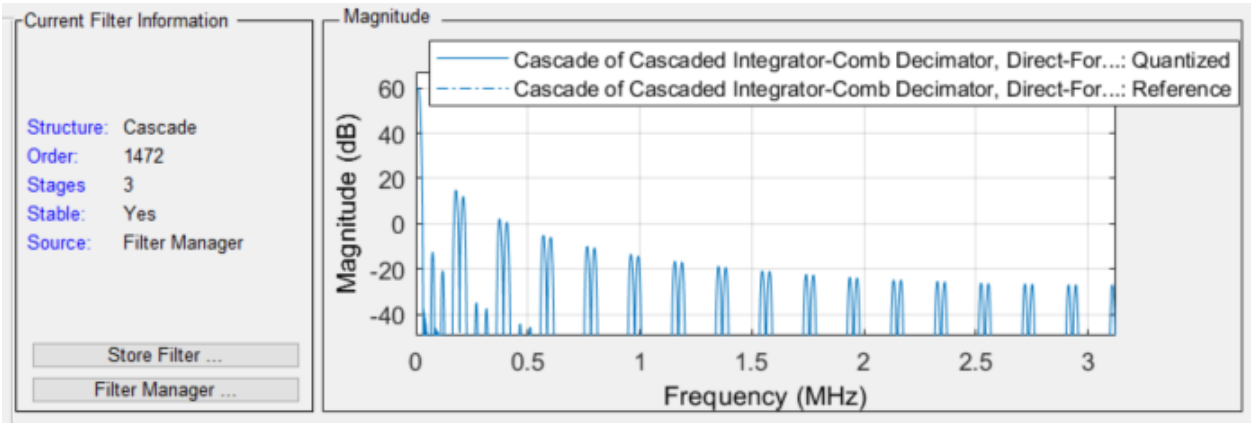


Figure 36: Cascaded filter response - Simulink

In figure 37 the complete simulink model is shown that was used for doing the simulations. In this overview it can be seen that the input signal is first given to the 2nd Order Noise Shaper. This Noise Shaper is the simulink implementation of the analog designed noise shaper. The simulink implementation overview is shown in figure 38. After the noise shaper 3 different filters are shown. One optimal filter that directly uses the output of the noise shaper is represented in the red square. This optimal filter was used as reference filter during the design phase. Also there are two quite similar filters. These both consist of a first filter stage. Subsequently this is followed by two FIR filter stages as can be seen in the yellow and green rectangles. The FIR filters in the green rectangle are optimized and the FIR filters in the yellow rectangle are for testing purposes.

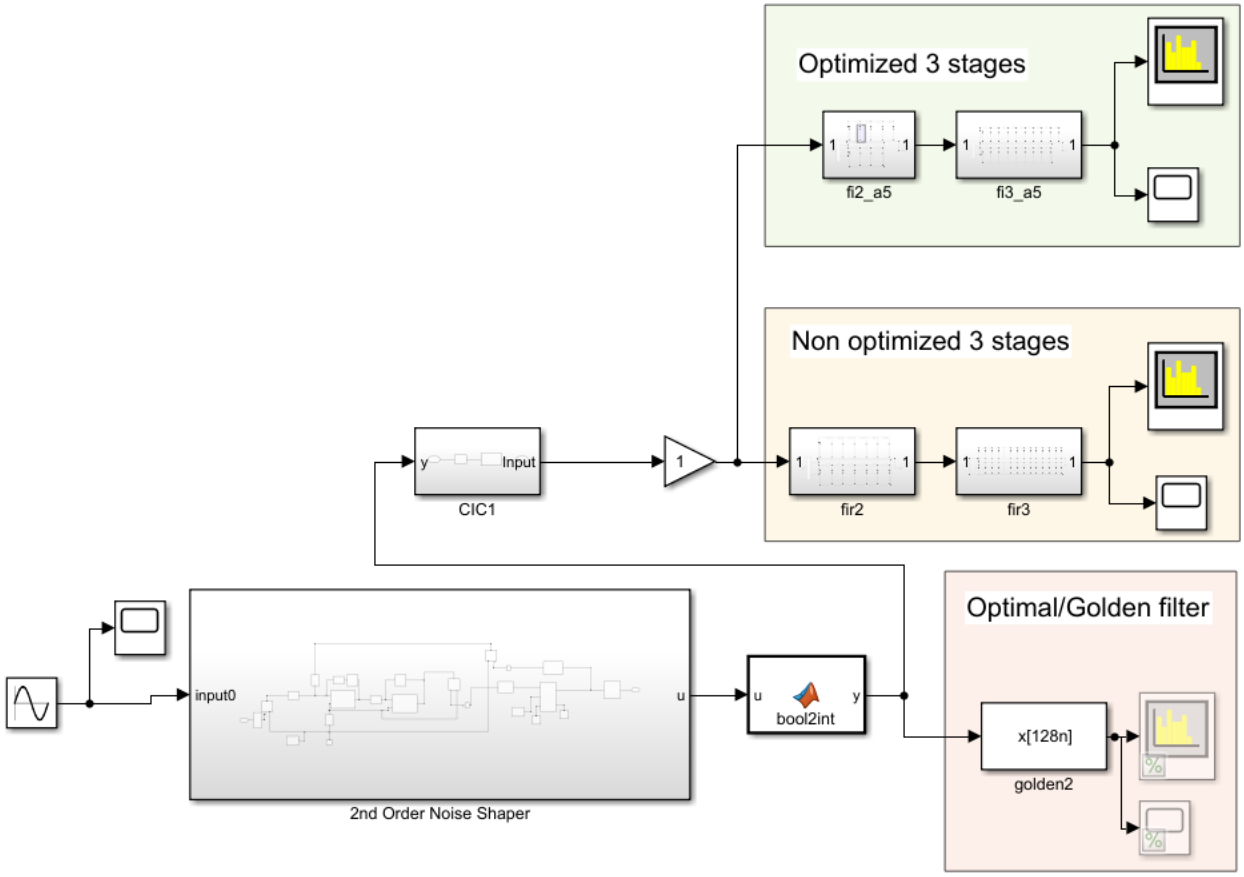


Figure 37: Complete simulink design overview

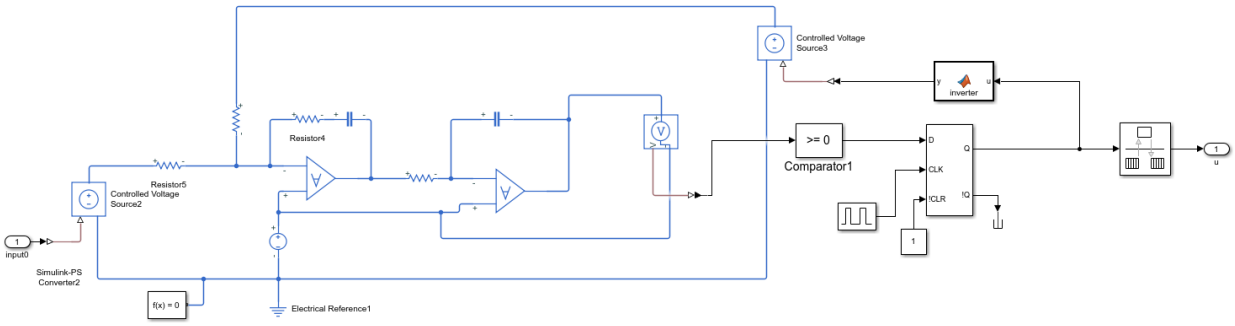


Figure 38: 2nd order noise shaper - Simulink

Table 2: Overview hardware

	Reference FIR	CIC stage 1	FIR stage 2	FIR stage 3	Sum hardware 3 stages
Filter order	2115	-	8	18	-
Decimation factor	128	32	2	2	-
Number of multipliers	2115	0	9	19	28
Number of adders	2114	4	8	18	30

4.4.2.5 Simulation results

Simulations have been performed for the 2nd order noise shaper, the refrence filter an the 3 stage filter.

The simulation results of the 2nd order noise shaper are shown in figure 39. In this plot both the input and output of the noise shaper are plotted. The yellow signal is the input and the blue signal is the output.

The simulation output of the reference filter is shown in figure 40 and the output of the 3 stage filter in figure 41. It should be noted that the amplitude of the 3 stage filter is scaled compared to the signal of the reference filter. The scaling of the 3 stage filter is caused by the first CIC filter stage

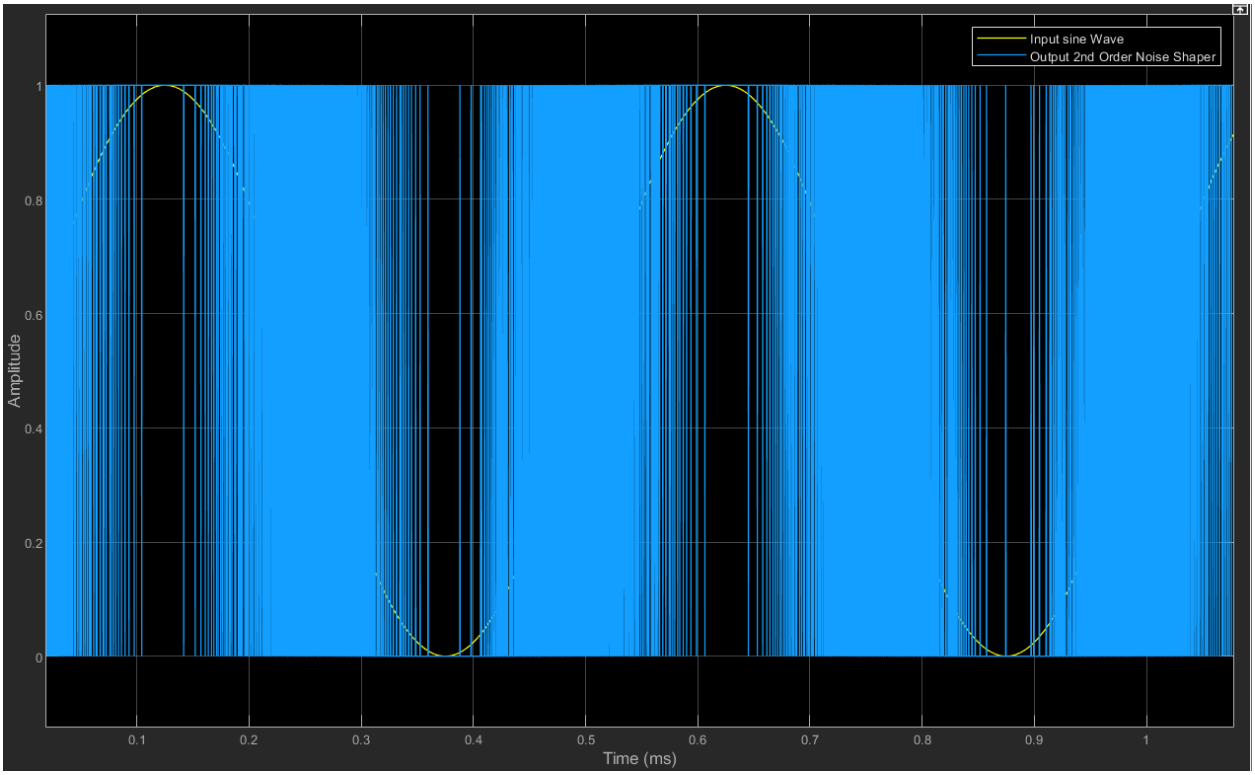


Figure 39: 2nd order noise shaper - Simulink simulation

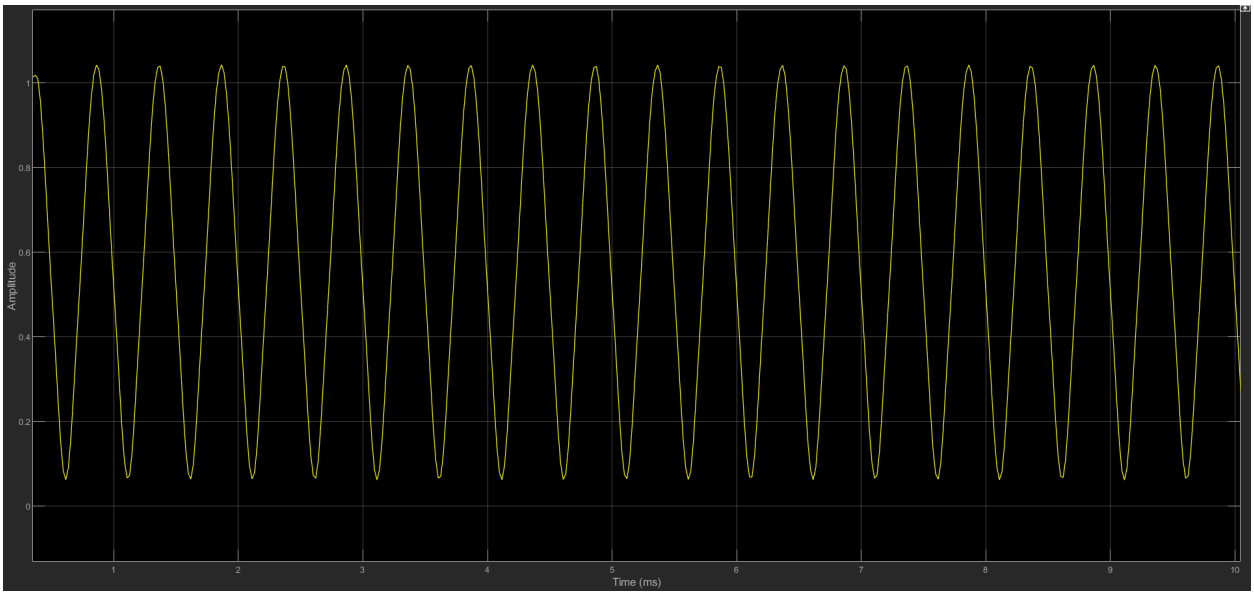


Figure 40: Simulation Reference filter

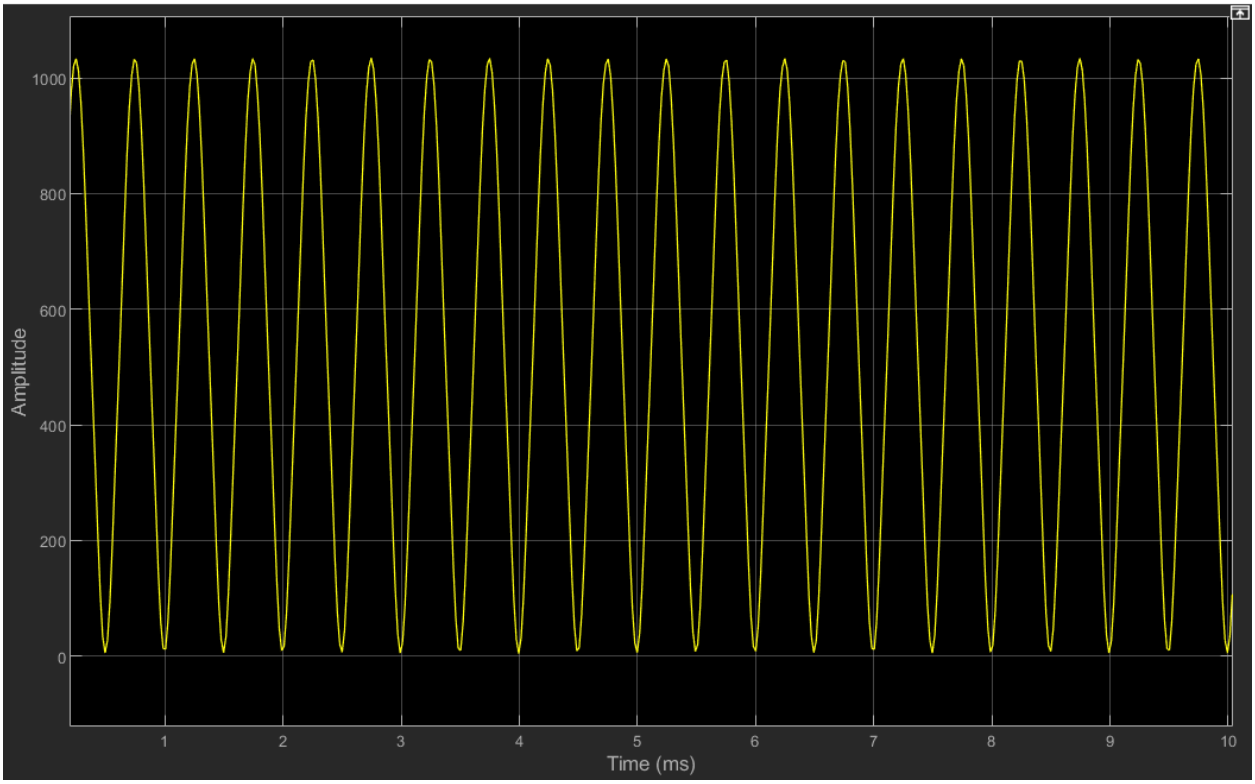


Figure 41: Simulation 3 stage filter

4.4.2.6 Signal to Noise Ratio

Using the presented Simulink model it was possible to determine the power-spectrum of the output signal of the 3 stage filter. This power-spectrum is shown in figure 42. This powerspectrum was used to determine the Signal to Noise Ratio and accordingly adjust the filters to meet the required specifications. The SNR can be calculated using equation 25. Doing this calculation for the power spectrum in figure 42 results in an SNR of 70dB

In this report only this theoretical SNR calculation will be shown. No SNR calculation has been performed on the FPGA, because due to COVID measures a separated analog and digital demo was realized. This demo will explained in more detail later in the report.

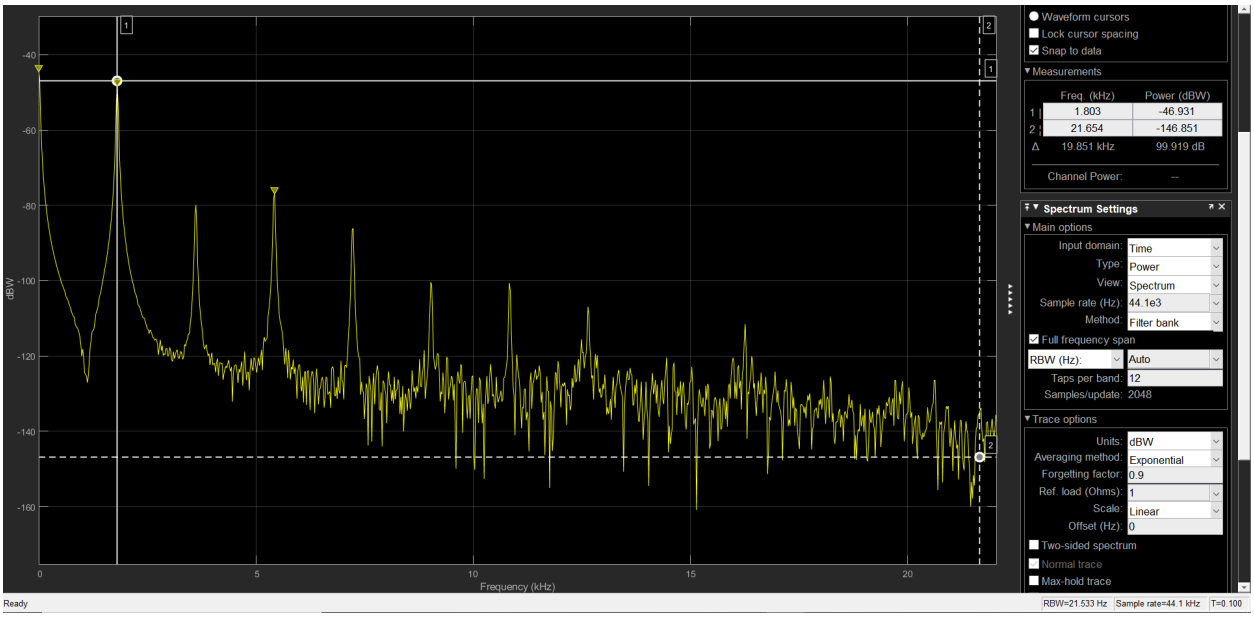


Figure 42: SNR cascaded filter

$$SNR = 10 * \log\left(\frac{\text{Max frequency}}{\text{RBW}}\right) + \text{dBW\_min} - \text{dBW\_max} \tag{25}$$

4.5 Realization

The models created and tested in Simulink have been converted to a schematic low-level hardware representation for the different filter components. There was first started with the creation of smaller sub-components. Subsequently these sub-components were put together to form the filter stages. These stages were again put togheter to form the whole filter system. This approach is also This hardware representation is then converted to code in VHDL that can be implemented on the FPGA. This method has also been used in making the VHDL realization. There is a lot of repetition of certain units in the structure. That is why for coding it is more efficient and clearer to make components of these units in the VHDL code and then connect them or reuse them.

4.5.1 CIC

The hardware of the CIC filter unit consists of 2 cascaded integrator structures followed by a decimator and two cascaded comb structures. These are combined to form the final CIC hardware as is shown in figure 43.

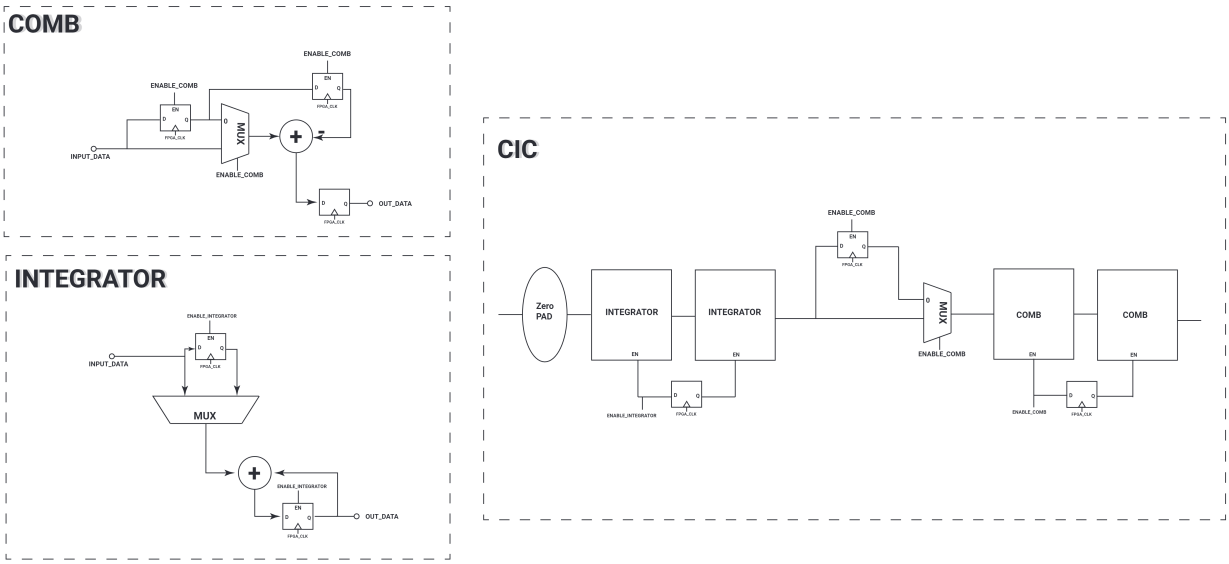


Figure 43: Hardware CIC

4.5.2 FIR

The hardware structure of FIR filter stage 2 and FIR filter stage 3 have been derived from the FIR structures as implemented in Simulink. Different architectures have been created for the different sub components of which the filter consists. These sub components are:

- Decimators combined in a decimation system
- Atom unit

4.5.2.1 Decimator

As is shown in figure 31 and figure 35 both stages of the FIR filter have a decimation system stage. This decimation system stage can be represented in hardware as is shown in figure 44. This hardware consists of two decimators with a decimation factor of 2. The bottom decimator is connected to a flipflop which functions as a delay element between the dataIN signal and the input of the bottom decimator. The decimator structure itself is shown in figure 45.

In the design of the FIR decimator there are 3 flipflops. The toggle flipflop is used to toggle the multiplexer that switches between new dataIN and old data that was previously given as dataOUT and is now stored in the data flipflop. A decimation factor of 2 implies that there is basically toggled between new dataIN and data that was already given as output to realize the desired decimation.

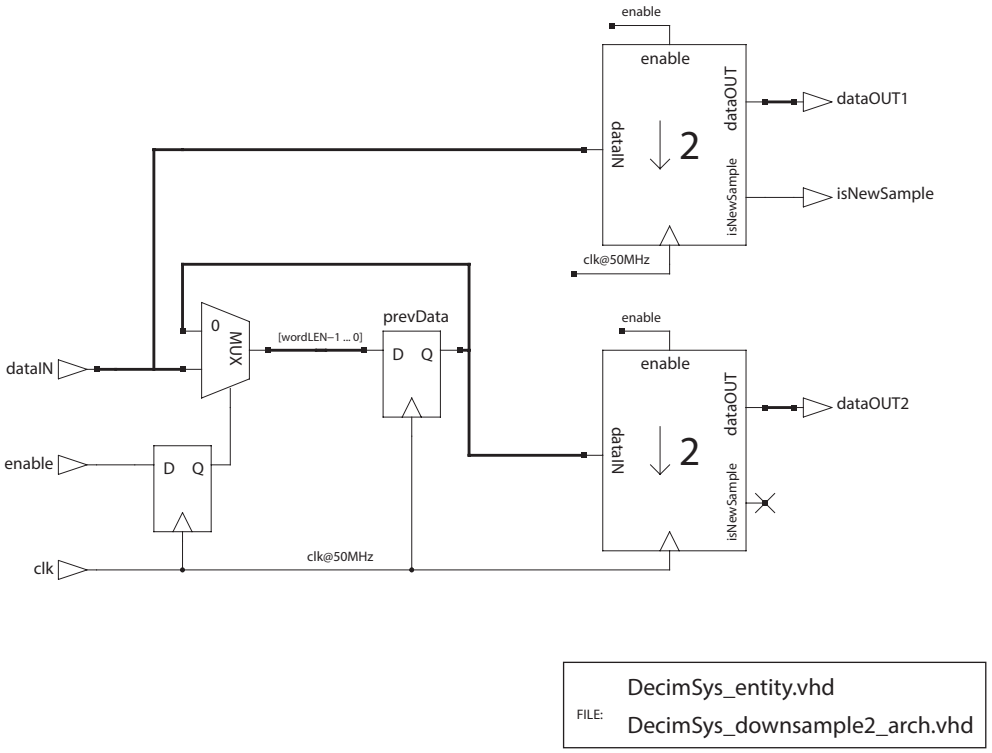


Figure 44: Hardware schematic FIR pre-stage

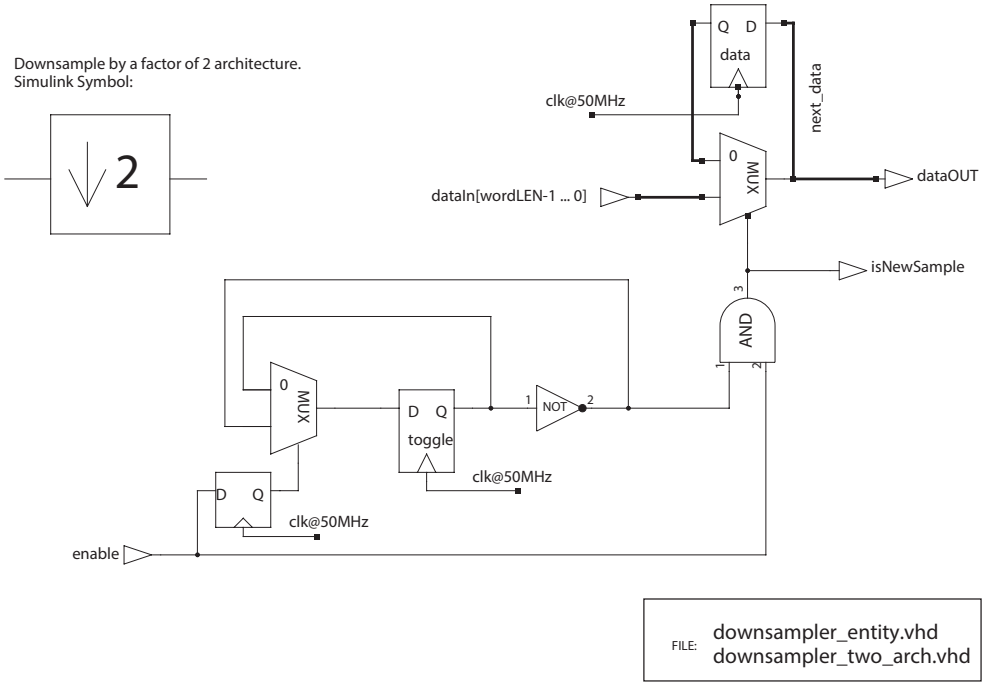


Figure 45: Hardware schematic FIR decimator

4.5.2.2 Atom unit

The FIR atom unit is implemented by means of a state machine. An overview of the different states and their relations is shown in figure 46. There has been chosen for an implementation by means of a state machine, because this would allow for computations in an efficient amount of clock cycles. The multiplications can happen simultaneously for all the FIR atom units, however the atoms should wait for each other on the result of the addition. By doing the multiplications at the same moment also the efficiency in time is increased. The structure of the atom units was programmed in a python file, which made it easy to generate the VHDL by combining the different atoms to form the structure of the FIR filters.

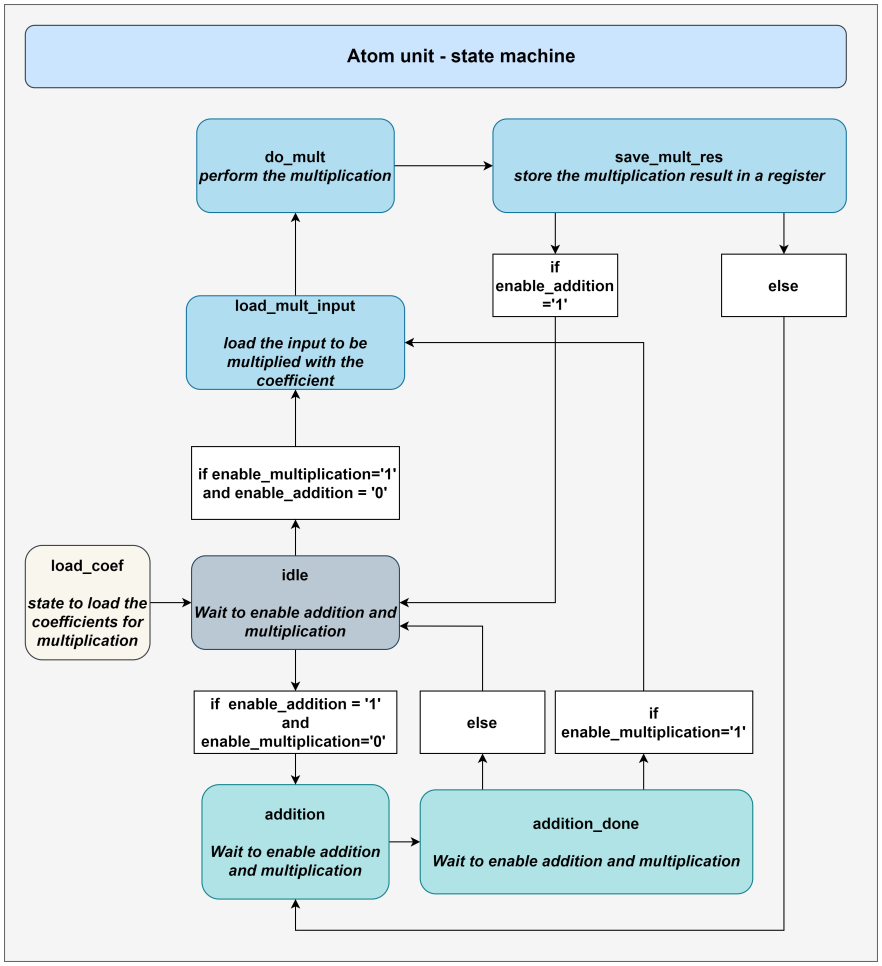


Figure 46: State machine FIR atom

4.5.2.3 Hardware overview

A schematic of the hardware of the complete FIR stage 2 implementation is shown in figure 47. The filter atoms are preceded by the decimation stage. Flip flops are used as delay elements and the output of the flip flops is given as input for the different FIR atom units. It should be noted that two FIFO structures are used in the FIR architecture. These FIFO structures contain hard coded filter coefficients that are loaded in the FIR atoms in the initialization stage upon using the filter. A simulation of loading the filter coefficients in the atoms is shown in figure 48. After the initialization stage the FIFO is used for its main purpose, which is as delay elements that first in first out pass the input data to the different atom units. Independently the

filter order, each FIR architecture contains two FIFO memories indicated as `fifo1` and `fifo2`.

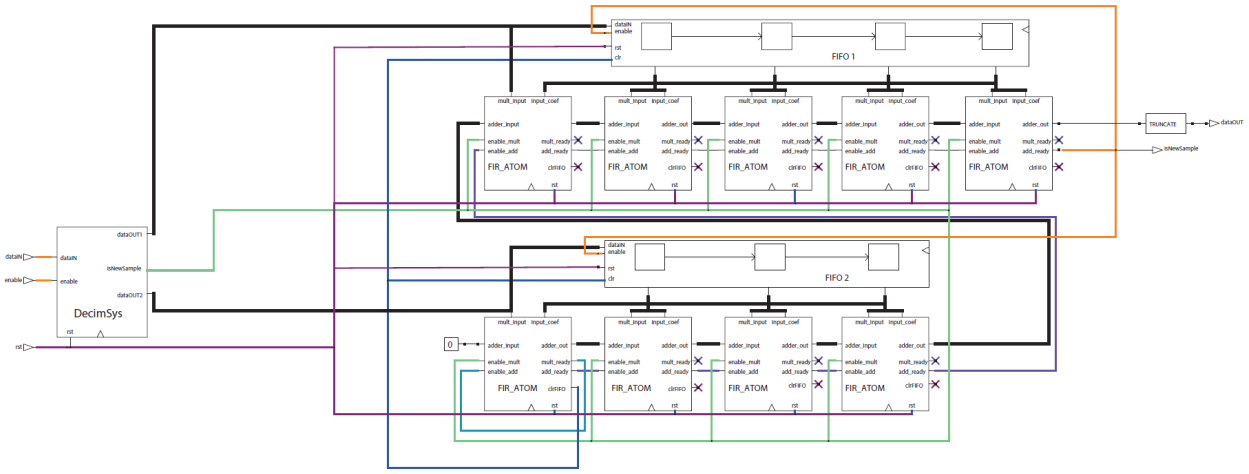


Figure 47: Hardware FIR stage 2

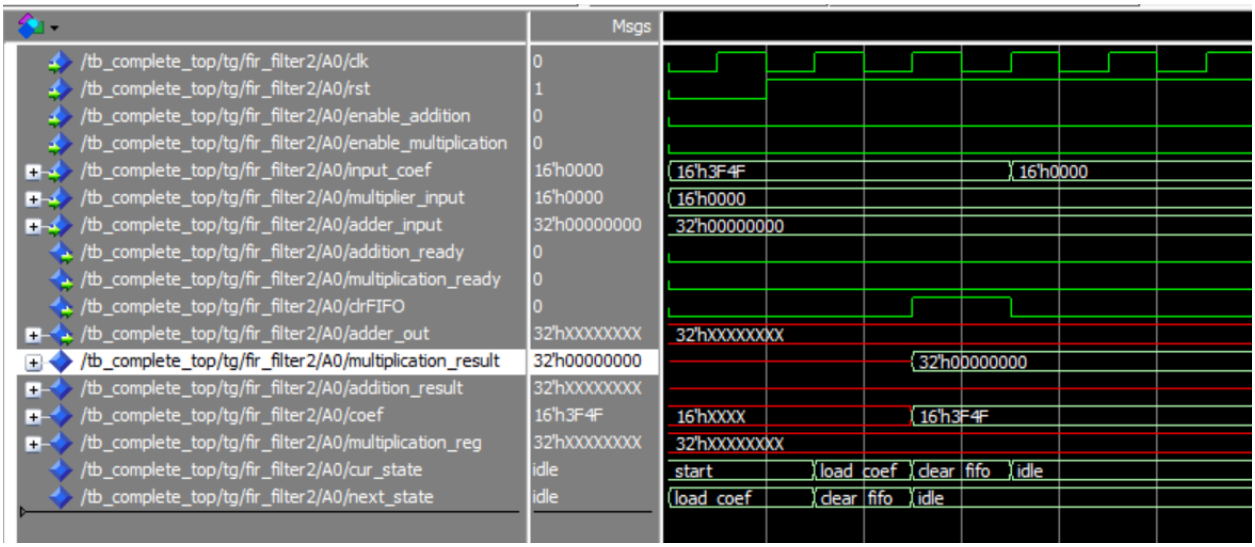


Figure 48: Simulation loading atom coefficients

4.5.3 Complete filter

The functional simulation result of the filter core (CIC + FIR + FIR) are shown in Figures 49, 50 and 51.

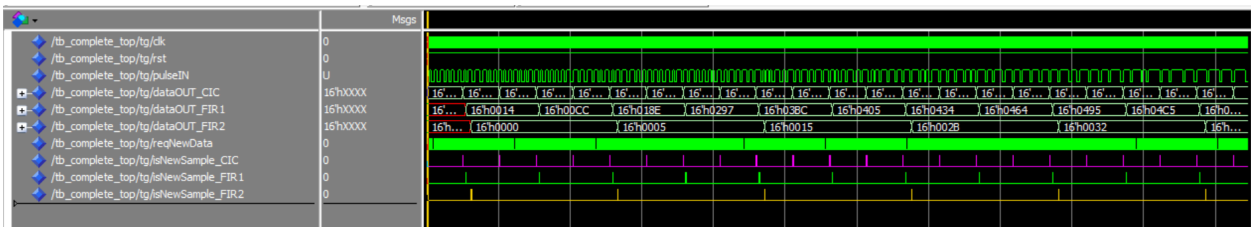


Figure 49: Decimation simulation 3 stages

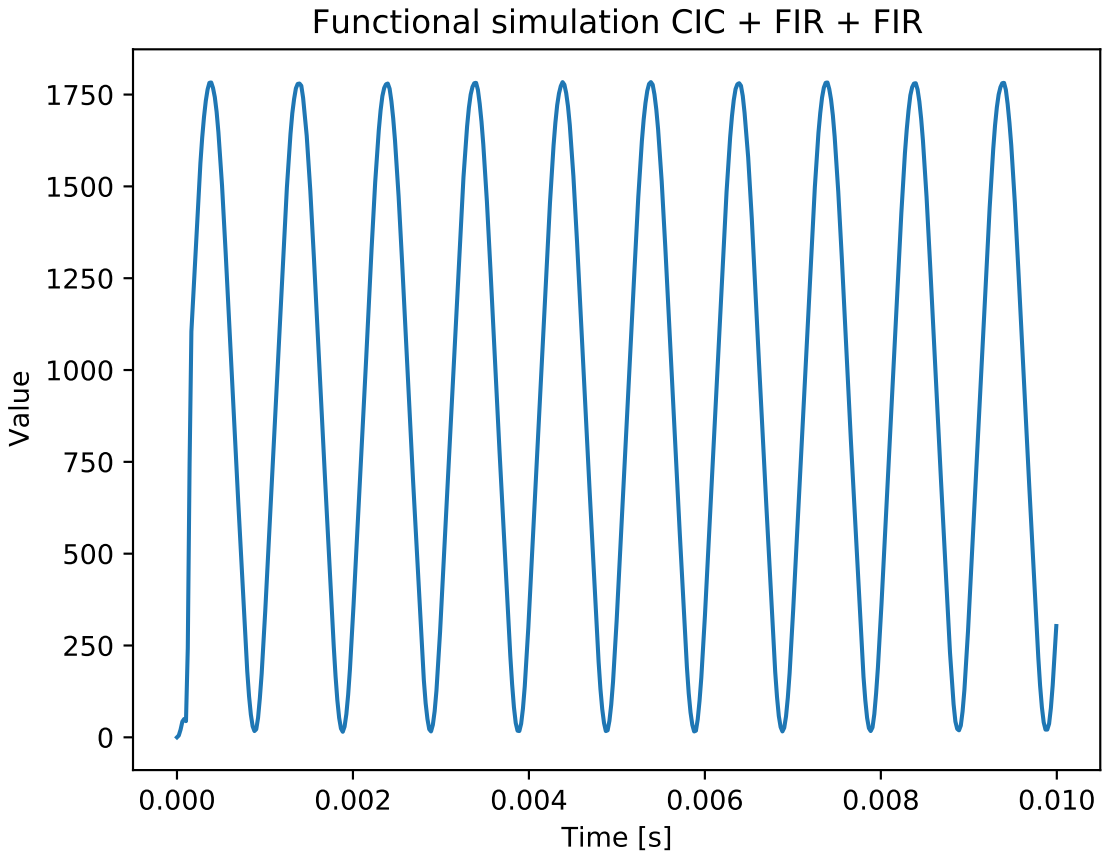


Figure 50: Functional simulation output. 1kHz input sinusoid.

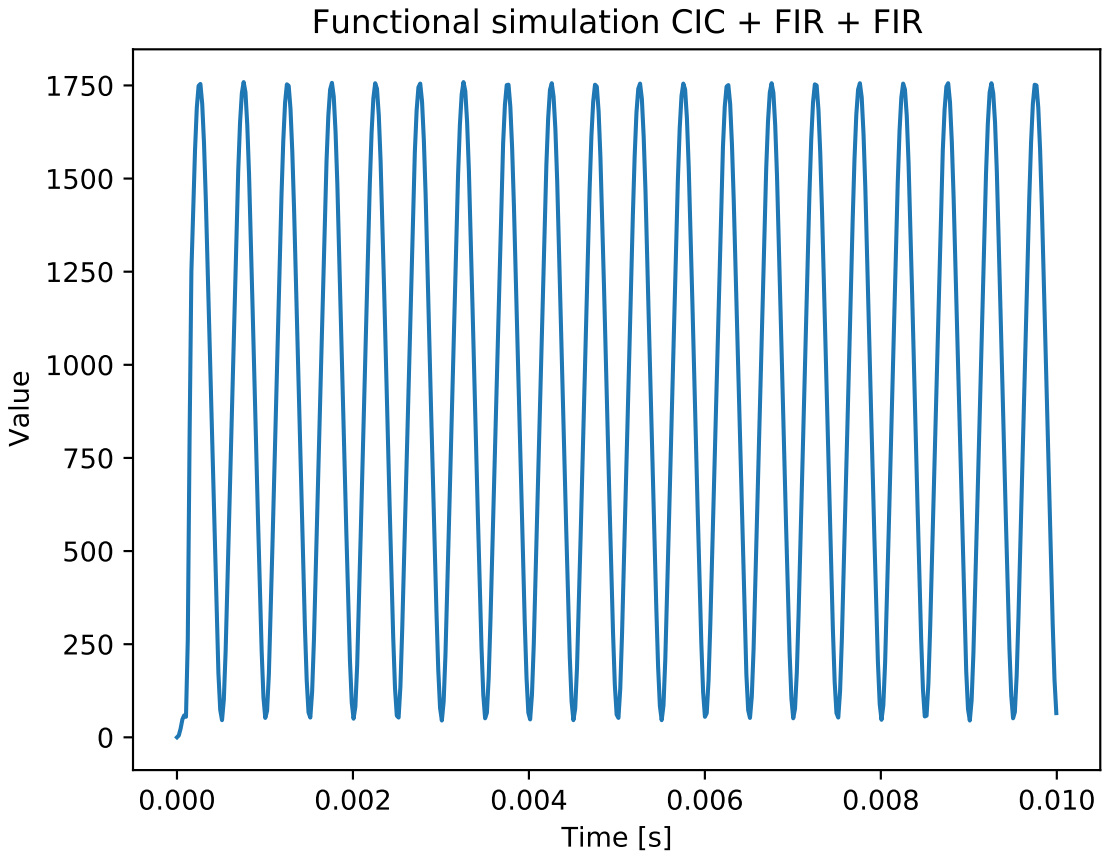


Figure 51: Functional simulation output. 2kHz input sinusoid.

4.5.4 Hardware in the loop

This project was carried out during the COVID-19 pandemic. In order to create a setup with which the entire system could be tested at home, it was decided to go the extra mile by realizing a hardware in the loop implementation. This hardware in the loop implementation implies that a loop is created for the information stream in which the digital filter is included.

A schematic of the steps that are included in the loop is shown in figure 52. The output data of the noise shaper modelled in Simulink and shown in figure 39 is used as input data for the FPGA. This is done by first sending the output of the simulation to a signal generator. The signal generator can not directly be connected to the FPGA. Therefore an interface had to be developed to enable such a connection. This interface has been realized be means of a protoshield as shown in figure 53. This protoshield forms a connection between the output of the signal generator and the input of the DE1 SOC FPGA. The signal generator is connected to a COAX connection point that is soldered on the protoshield board. Connections from the COAX connection have been etched on the protoshield board to form a connection with a female pin header block for connection to the FPGA and to header blocks for the ground. These different units have been highlighted in figure 53. The data given as input to the FPGA as given as input to the filter. The output of the filter is stored in a buffer. Using a serial TX communication interface the buffer data can be read on the PC. Eventually the data can be plotted and the loop is closed.

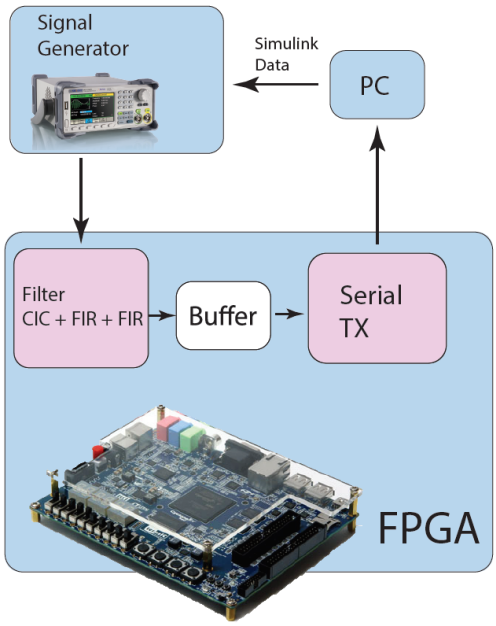


Figure 52: Hardware in the loop schematic

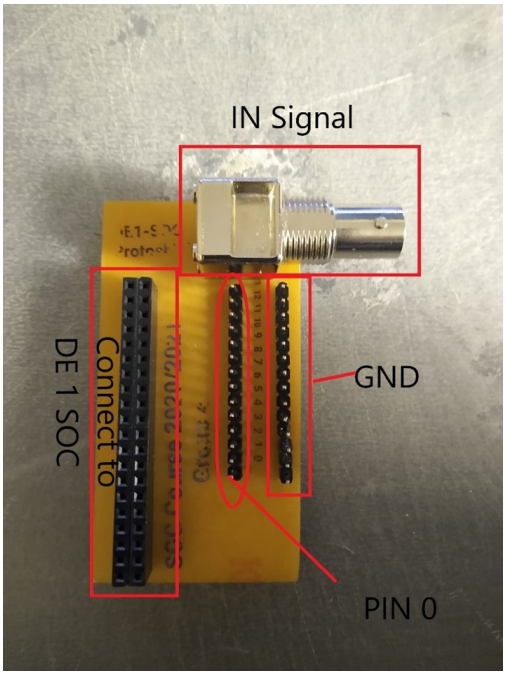


Figure 53: Proto shield

4.5.5 Synthesis

The FPGA used for the synthesis is a Cyclone 5 5CSEMA531C6. In figure 54 a flow summary of the synthesises on this FPGA is shown. The flow status is successful. Also the amount of registers, memory bits

and DPS blocks are shown. In figure 55 the pin plan is shown that was used on the FPGA. It was important to specifically choose the correct pins in order to connect the proto shield and the serial TX interface to the FPGA. In the synthesis compilation report also the worst-case slack was calculated, which had a value of 13.271 ns. This is also shown in figure 56.

Flow Summary

<<Filter>>

Flow Status	Successful - Wed Jan 13 15:39:30 2021
Quartus Prime Version	18.0.0 Build 614 04/24/2018 SJ Standard Edition
Revision Name	sdDemo
Top-level Entity Name	filter4sigmaDelta
Family	Cyclone V
Device	5CSEMA5F31C6
Timing Models	Final
Logic utilization (in ALMs)	807 / 32,070 (3 %)
Total registers	1879
Total pins	6 / 457 (1 %)
Total virtual pins	0
Total block memory bits	4,096 / 4,065,280 (< 1 %)
Total DSP Blocks	28 / 87 (32 %)
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	0 / 6 (0 %)
Total DLLs	0 / 4 (0 %)

Figure 54: Flow report FPGA demo

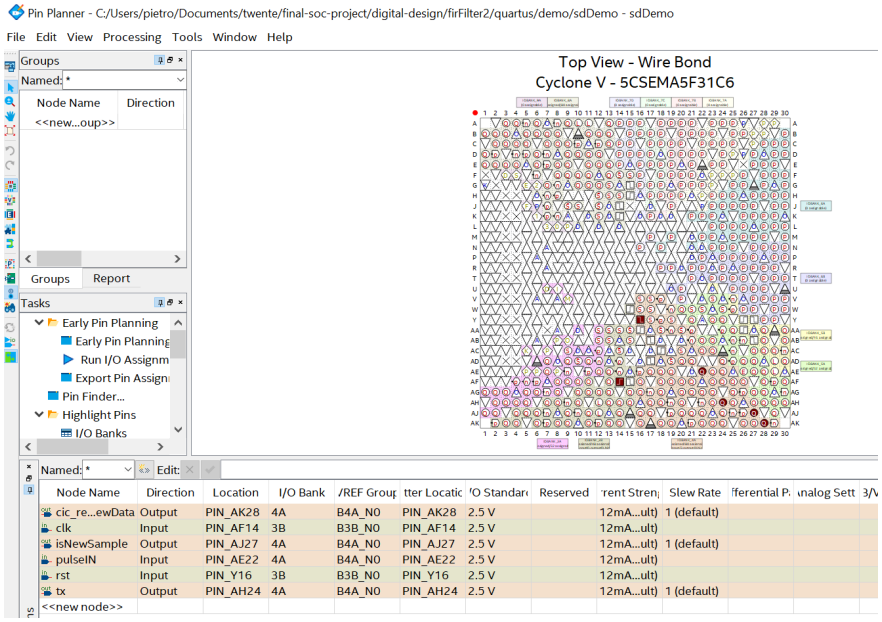


Figure 55: Pin plan used on the FPGA

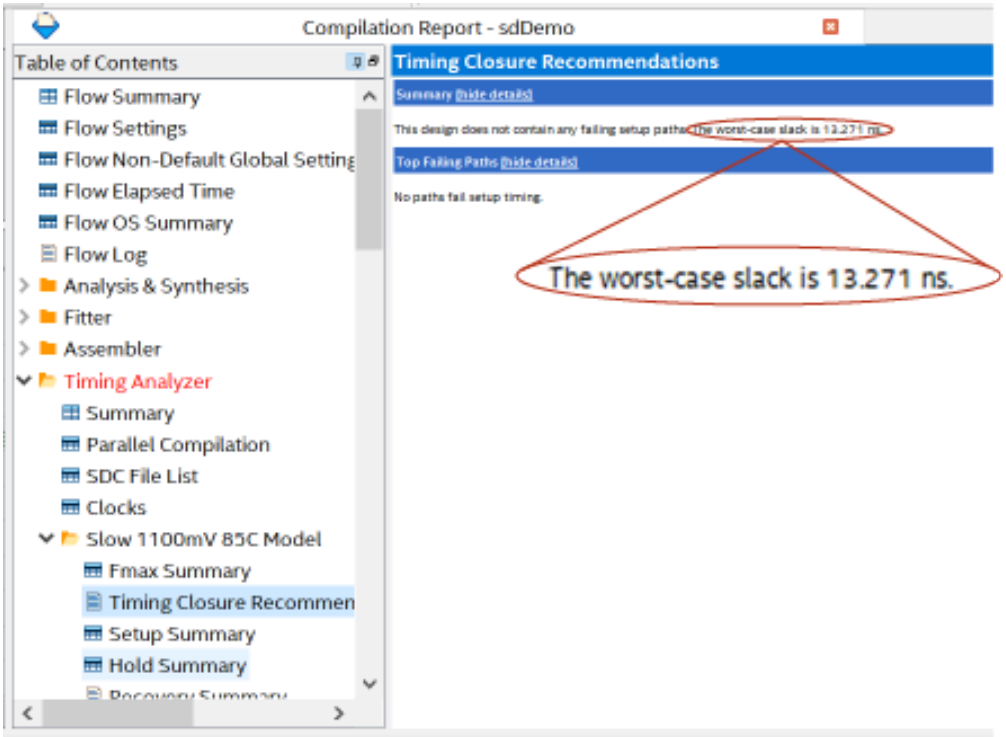


Figure 56: Slack of synthesised 3 stage filter

4.5.5.1 RTL CIC

As a result of the synthesis of the design in Quartus it is possible to create the RTL diagrams of the hardware that is assigned for the different designed hardware units. It is interesting to look at these automatically created diagrams, because they can be compared to the hardware schematics that were used to code the VHDL. In figure 57 the integrator hardware RTL is shown. Figure 57 shows the comb structure that follows the integrators. The complete RTL overview of the CIC filter combining the integrator and comb is shown in figure 59.

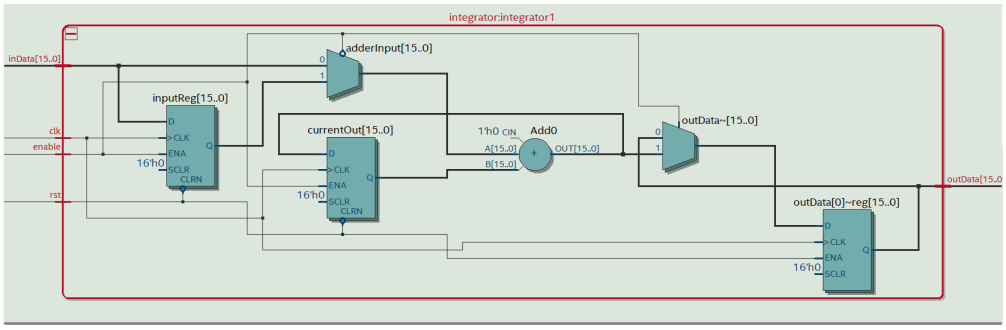


Figure 57: Integrator RTL view

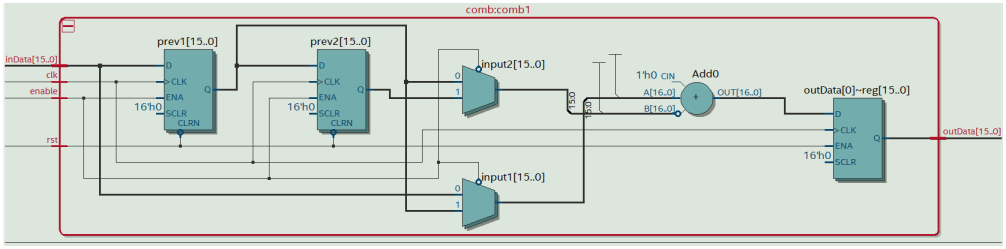


Figure 58: Comb RTL view

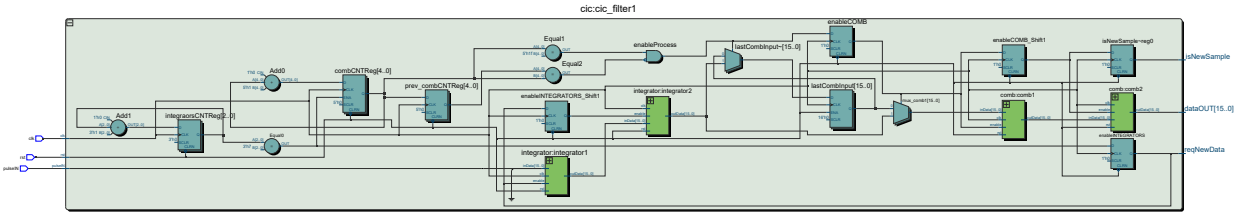


Figure 59: Complete CIC RTL view

4.5.5.2 RTL FIR

Similarly as for the CIC filter also the RTL overview for the synthesised FIR filter structures could be created. In figure 60 the RTL overview of the filter atom is shown. The complete RTL overview of the complete FIR stage 1 filter is shown in figure 61. Only the RTL of FIR stage 1 is shown in this report. The RTL of FIR stage 2 is very similar, but larger because of the greater number of FIR atoms used.

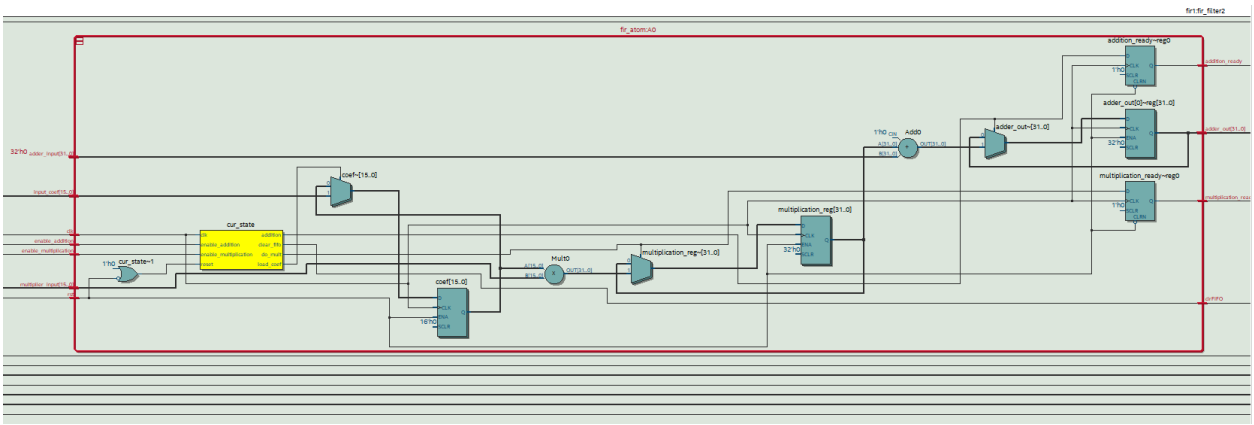


Figure 60: Filter atom RTL view

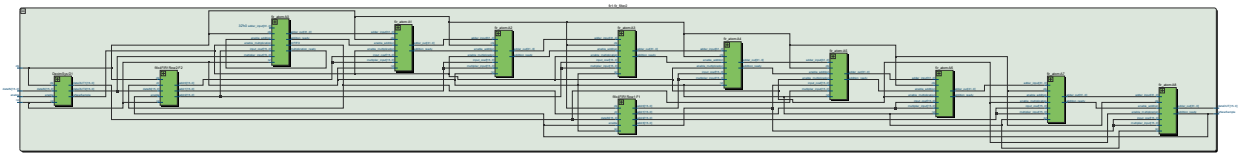


Figure 61: Complete FIR stage 1 RTL view

4.5.5.3 RTL Total design

In figure 62 an RTL overview is shown on how the hardware of the different filter stages is connected to each other. This overview also clearly shows that data flow through the filter structure. I can be seen how the data goes through the different filter structures to be temporarily stored in the buffer until it can be communicated to the PC using the uartTX interface. The specific RTL overview of the data buffer is shown in figure 63.

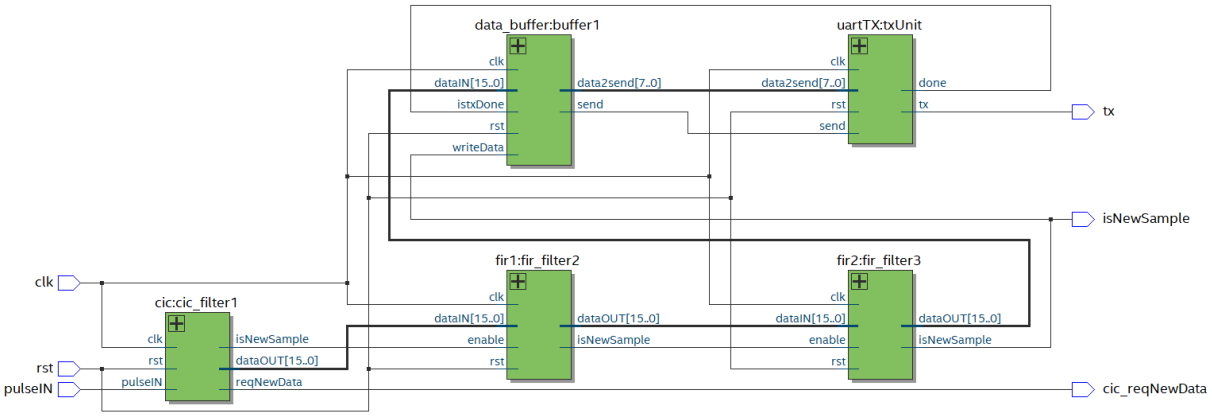


Figure 62: General RTL overview

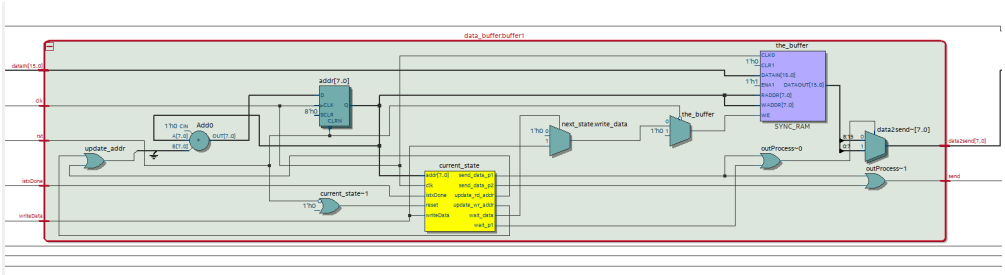


Figure 63: Data buffer

4.5.5.4 Post synthesis simulation

After the design was synthesized, a post synthesis took place. Because the Cyclone 5 FPGA is used, a time analysis was not supported for the post synthesis. However, it was possible to do a simulation with the post synthesis model. The post synthesis has only been performed for the cascade of the 3 different filter stages. As expected, this gave more or less the same results as the pre-synthesis simulations. The simulation results as presented in the Modelsim simulation windows is shown in figure 64. The wave form resulting from this simulation is shown in figure 65.

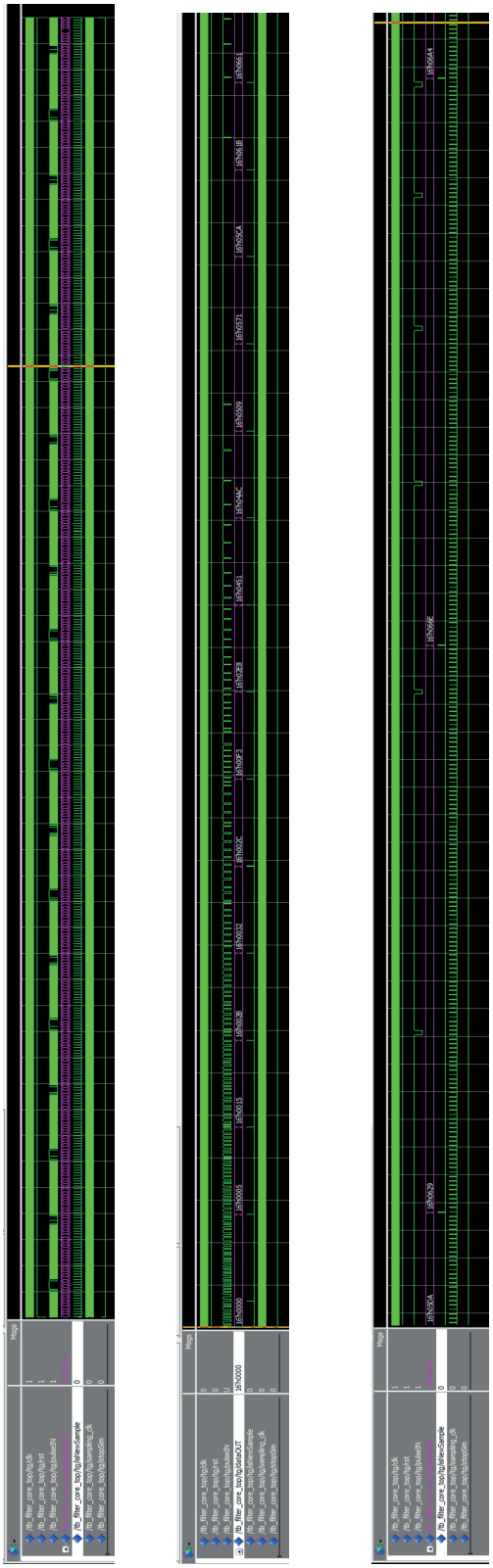


Figure 64: Functional Post Synthesis Simulation of the Digital Filter. 1 kHz input sinusoide.

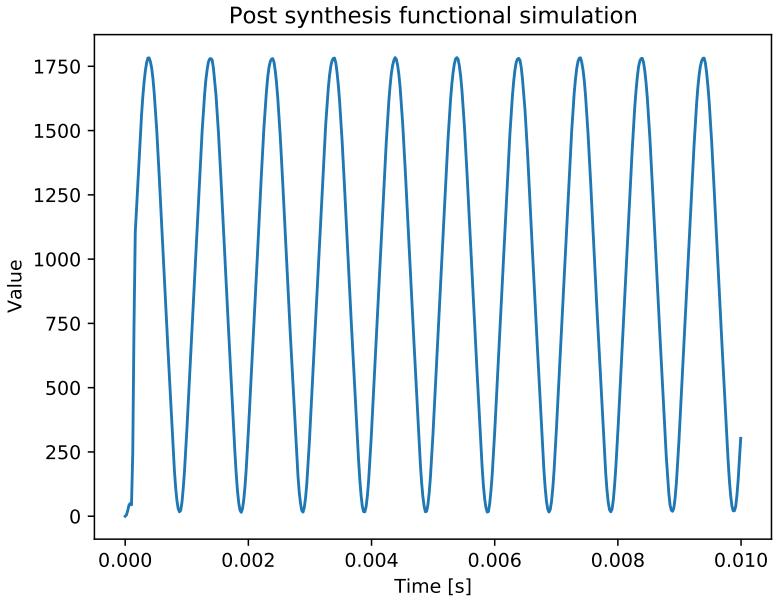


Figure 65: Post synthesis simulation. 1 kHz input sinusoide.

4.6 Measurement results

4.6.1 CIC stage 1 measurements

The demo of the CIC filter implementation together with an overview of the hardware in the loop incorporating this filter can be seen by using the following url and in figure 66:

`https://www.youtube.com/watch?v=OmDwLAizIj8&feature=youtu.be`.

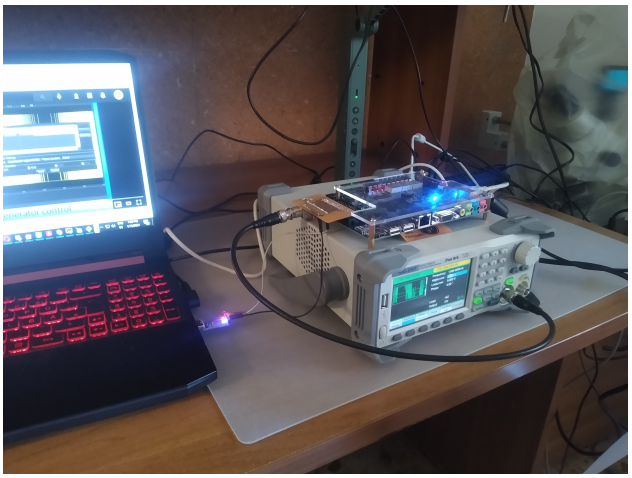


Figure 66: Hardware Setup

4.6.2 CIC stage 1 + FIR stage 2 + FIR stage 3 measurements

The same output signal as was produced by the noise shaper implemented in Simulink was provided to the FPGA. The resulting output signal of the FPGA is shown in figure 67 . The output is a clear sinusoid, similar in shape as the one that could be seen in the different simulations.

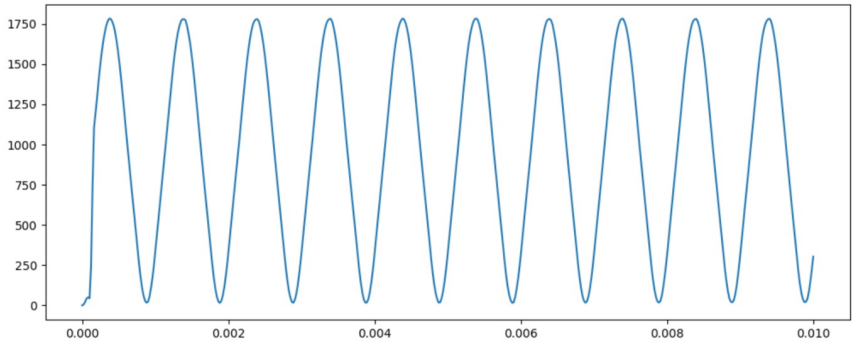


Figure 67: Measurement hardware in loop - CIC stage 1 + FIR stage 1 + FIR stage 3

4.7 Possible improvements

A possible improvement can be found in the architecture of the filter atoms. Not all atom additions are used at the same time. Therefore instead of the structure currently used a structure with parallel multipliers could be implemented together with one adder and a register to store the output. The computation of this structure will take more clock cycles then the current configuration, however it will take up less area. It is therefore a typical computation time - area tradeoff.

4.8 Digital conclusion

A 3 stage digital filter has been investigated, designed and realized for implementation in a sigma delta modulator. A realistic Simulink model has been created and evaluated for the whole hardware setup. A cascaded filter has been created using one CIC filter stage and 2 subsequent FIR filter stages. It can be concluded that it is possible to realize a fully functioning hardware in the loop implementation using this filter. From measurements it can therefore been concluded that the investigation, designing and realization phases have led to a very successful result. It has been shown that the output of the Simulink noise shaper model can effectively be implemented in the hardware in the loop setup to mimic the signal created in a real cascade of the analog and digital part of the sigma delta modulator. Therefore it can also be concluded that despite the limitations imposed by the corona restrictions present during the whole project, a more than realistic setup has been made of the digital part of the sigma delta modulator showing the capabilities of the designed filter.

5 Delivered `vhdl` and script files

- Files for the CIC Filer
  - `./vhdl/cic_entity.vhd`. Entity declaration of the CIC filter.
  - `./vhdl/cic_n2m1.vhd`. Architecture of the CIC Filter.
  - `./vhdl/comb_entity.vhd`. Comb filter entity.
  - `./vhdl/comb_standardComb_arch.vhd`. Architecture of the Comb filter.

  - ./vhdl/integrator_entity.vhd.
  - ./vhdl/integrator_standardIntegrator_arch.vhd.

- Files for the FIR filter 1 & 2
  - ./vhdl/DecimSys_entity.vhd
  - ./vhdl/DecimSys_downsample2_arch.vhd
  - ./vhdl/downsampler_entity.vhd
  - ./vhdl/downsampler_two_arch
  - ./vhdl/fifo4FIR1Row1_entity.vhd [1]. Entity declaration of the fifo1 used for the FIR 1.
  - ./vhdl/fifo4FIR1Row1_s5WL16_arch.vhd. Architecture of the fifo1 used for the FIR 1.
  - ./vhdl/fifo4FIR1Row2_entity.vhd. Entity declaration of the fifo2 used for the FIR 1.
  - ./vhdl/fifo4FIR1Row2_s4WL16_arch.vhd. Architecture of the fifo2 used for the FIR 1.
  - ./vhdl/fifo4FIR2Row1_entity.vhd. Entity declaration of the fifo1 used for the FIR 2.
  - ./vhdl/fifo4FIR2Row1_s10WL16_arch.vhd. Architecture of the fifo1 used for the FIR 2.
  - ./vhdl/fifo4FIR2Row2_entity.vhd. Entity declaration of the fifo2 used for the FIR 2.
  - ./vhdl/fifo4FIR2Row2_s9WL16_arch.vhd. Architecture of the fifo2 used for the FIR 2.
  - ./vhdl/fir_atom_entity.vhd. Entity declaration of the FIR atom.
  - ./vhdl/fir_atom_atom_arch.vhd. Architecture of the FIR atom.
  - ./vhdl/fir1_entity.vhd. Entity declaration of the first fir filter.
  - ./vhdl/fir1_fa9_arch.vhd. Architecture of the first FIR filter.
  - ./vhdl/fir2_entity.vhd. Entity declaration of the second FIR filter.
  - ./vhdl/fir2_fa19_arch.vhd. Architecture of the second FIR filter.

- Digital filter core
  - ./vhdl/filter_core_cicFirFir_arch.vhd. Entity declaration of the filter core.
  - ./vhdl/filter_core_entity.vhd. Architecture of the filter core.

- vhdl files developed for the demo
  - ./vhdl/fpga/buffer_entity.vhd
  - ./vhdl/fpga/buffer_fill_and_empty_arch.vhd
  - ./vhdl/fpga/filter4sigmaDelta.vhd. Entity declaration and architecture of the fpga demo (Filter Core + Buffer + UART).
  - ./vhdl/fpga/uartRec_entity.vhd. UART receiver entity declaration. Note this file has been developed for a previous project, thus some of the vhdl best practices introduced during the course were not applied.
  - ./vhdl/fpga/uartRec_rec_arch.vhd. UART receiver architecture. Note this file has been developed for a previous project, thus some of the vhdl best practices introduced during the course were not applied.
  - ./vhdl/fpga/uartTX_entity.vhd. UART transmitter entity. Note this file has been developed for a previous project, thus some of the vhdl best practices introduced during the course were not applied.

[1] fifo4FIR1Row1_entity.vhd, fifo4FIR1Row2_entity.vhd, fifo4FIR2Row1_entity.vhd and fifo4FIR2Row2_entity.vhd contain the same vhdl code

  - `./vhdl/fpga/uartTX_tx_arch.vhd`. UART transmitter architecture. Note this file has been developed for a previous project, thus some of the `vhdl` best practices introduced during the course were not applied.

- Files used for the `vhdl` test bench
  - `./vhdl/tb/fileIO.vhd`. Component used for simulation purposes only. Its function is to read the input of the filter core from a text file and write the input to a text file.
  - `./vhdl/tb/tb_filter_core.vhd`. This testbench is used to test the filter core.
  - `./vhdl/tb/tb_filter4sigmaDelta.vhd`. This testbench is used to test the cascade of the filter core, the buffer and the UART tx unit.

- Configuration Files
  - `./vhdl/config/conf_tb_filter_core.vhd`. Configuration used to test the filer core (functional simulation).
  - `./vhdl/config/conf_tb_filter_core_post.vhd`. Configuration used to test the filer core (post synthesis functional simulation).
  - `./vhdl/conf_tb_filter4sigmaDelta`. Configuration used to test test the fpga demo (functional simulation).

- Output Synthesis File
  - `./vhdl/filter_core.vho`. This file is generate by `Quartus` software for the `Cyclone V 5CSEMA5F31C6` fpga. This file **does not** contain the UART transmission unit used for the demo.
  - `./vhdl/fpgaDEMO.vho`. This file is generate by `Quartus` software for the `Cyclone V 5CSEMA5F31C6` fpga. This file is the synthesis of the filter core, the buffer and the UART tx.

- `python` scripts
  - `./python/soc_vhdl_utils/soc_vhdl_utils.py`. This script contains all the function used to generate the `vhdl` code for the FIR filters and FIFO memories.
  - `./python/genFIR/genFIR.py`. This script generates the `vhdl` code for the FIR filters and FIFO memories.
  - `./python/read_and_plot/read.py`. This script read the data sent by the fpga to the serial port.
  - `./python/read_and_plot/plot.py`. This script plot the data received from the fpga.

- Matlab/Simulink files
  - `./matlab/ADC3stages_filters.slx`. Simulink model used to simulate the entire system (noise shaper + digital filter).

References

[1] *The signal path, Sigma-delta modulation, www.TheSignalPath.com.*

[2] R. J. Baker, *CMOS Mixed-Signal Circuit Design, Second Edition.* Wiley, 2008.

[3] S. Park, "Principles of sigma-delta modulation for analog-to-digital converters," 1999.

[4] R. Abbiati, A. Di Odoardo, A. Geraci, and G. Ripamonti, "A programmable a/d sigma-delta converter for pulse digital processing setups," *IEEE Transactions on Nuclear Science*, vol. 51, no. 3, pp. 1270–1276, 2004.

[5] M. Donadio, "Cic filter introduction," 08 2000.

[6] S. Pavan, R. Schreier, and G. C. Temes, *Understanding delta-sigma data converters.* John Wiley & Sons, 2017.

[7] M. Fowler, "Polyphase filters - a model for teaching the art of discovery in dsp," 06 2011, pp. 2908 – 2911.

[8] K. Mohammed, "Design and implementation of decimation filter for 15-bit sigma-delta adc based on fbga," 02 2018.