Wireless Communication Systems

Filtering a 5G channel

1 Introduction

The purpose of this lab is to continue the study that we have started in the filtering Lab. The objective is to optimize the baseband part of a 5G receiver. The system and electrical simulations will be carried out respectively on Octave/Matlab and on Cadence.

2 Scenario

Figure 1 shows a classical homodyne receiver. In this lab, we will just focus on the baseband part showed in the doted boxes which consists of :

- A low pass anti-alias filter (AAF)
- A variable Gain Amplifier (VGA)
- An Analog-to-Digital Converter (ADC)

For the sake of simplicity, we will just consider the 4 neighboring interferers of the useful signal. After the mixer, the signal at both channels I and Q will be as shown in Figure 1 :

- 0 to 10 MHz ⟶ The useful signal that folded on itself
- 10 to 30 MHz ⟶ Interfers B and C on the top of each other, the combination of the two will be called interferer 1
- 30 to 50 MHz ⟶ Interfers A and D on the top of each other, the combination of the two will be called interferer 2

Using the I and Q outputs shifted by 90 degrees, the 20 MHz useful signal is then reconstructed in digital using a phase shifter and a Hilbert filter. As said earlier, in this lab, we will just focus on the baseband part. The AAF is matched to 50 Ω and the signal at its input will be modeled as shown in Figure 2.

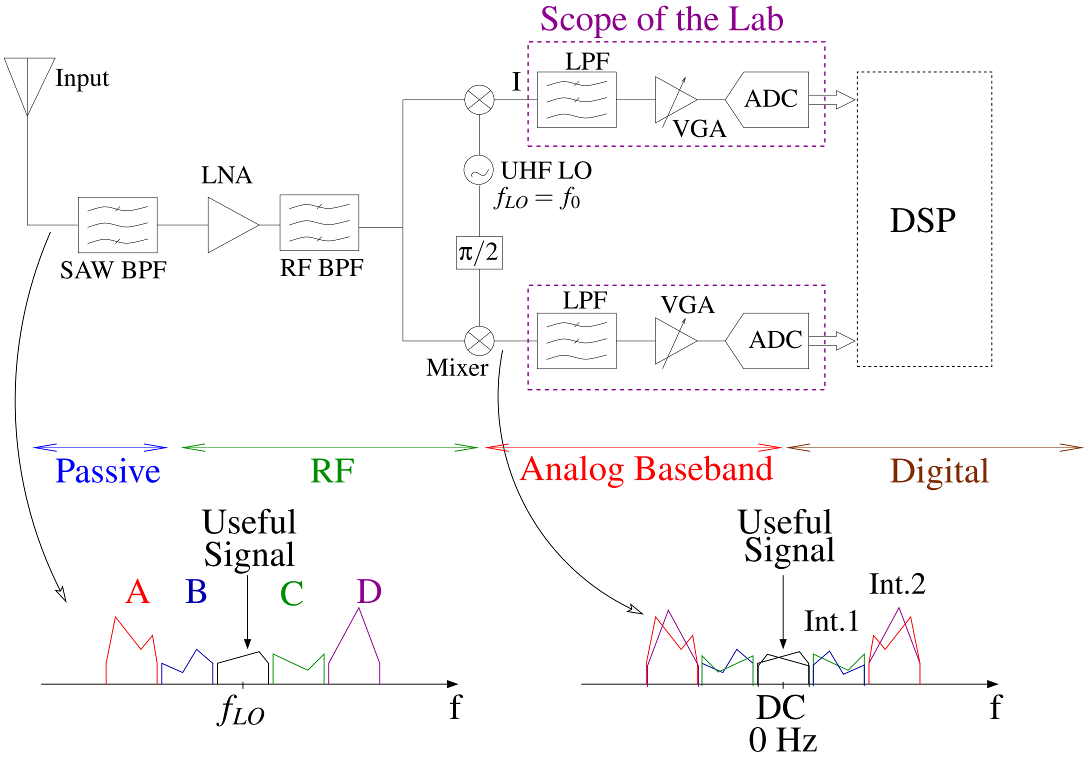

FIGURE 1 – I/Q homodyne Receiver

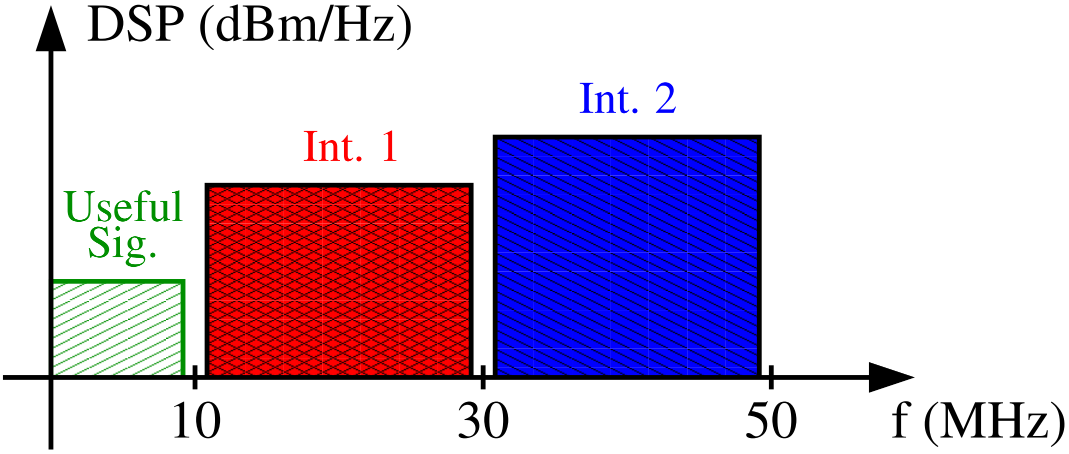

FIGURE 2 – Spectrum at the AAF input

In the filtering lab, we analyzed the impact of the co-optimization of the filtering and the ADC on the overall performance. We will continue this analysis by exploring the ADC architecture choice.

3 ADC Flash

As mentioned earlier, the targeted application is the mobile standards 5G. The ADC for this application should have a bandwidth of 10 MHz and a signal to noise ratio (SNR) greater than 72 dB. Moreover, since we are targeting a mobile application, the power consumption should be as low as possible.

We will first investigate the flash architecture to implement our ADC.

We remind that the SQNR[1] for an ADC is given by the following formula :

$$SQNR = 10 \log \left(\frac{P_{signal}}{P_{noise}} \right)$$

with

$$P_{signal} = \frac{A^2}{2} for\ a\ sine\ input \qquad\qquad P_{noise} = \frac{q^2}{12} = \frac{\left(\frac{FS}{2^{nb}}\right)^2}{12}$$

Where A is the signal amplitude, q the quantization step, FS the ADC Full Scale and nb the number of bits.
In this lab, the only considered noise source will be the quantization noise and therefore SQNR = SNR.

Question 3.1 *Establish the relationship between SNR and nb in dB and deduce the number of bits required for obtaining the targeted resolution for an input signal with an amplitude $FS/2$.*

Question 3.2 *Load the file `flash.m` into the editor of Matlab. Set nb to the value obtained in the previous question. Plot and analyze the output signal in time domain and in frequency also. Compare the practical SNR to the theoretical SNR.*

Question 3.3 *By relying on the course materials, discuss the disadvantages of such an implementation.*

To reduce the complexity of our ADC, one option would be to use oversampling. In other words, instead of sampling the signal at twice its bandwidth, in this case 20 MHz (2 × 10 MHz), we will sample the signal at OSR (oversampling ratio) times 20 MHz. This will spread the quantization noise over a wider frequency range (see Fig. 3). A low pass filter in the digital domain should be employed to eliminate noise for frequencies above Bw before decimating the signal. The SQNR expression becomes in the oversampling case

$$SQNR = 6.02nb + 1.76 + 10 \log(OSR) + 20 \log \left(\frac{2A}{FS} \right)$$

[1] Signal to Quantization Noise Ratio

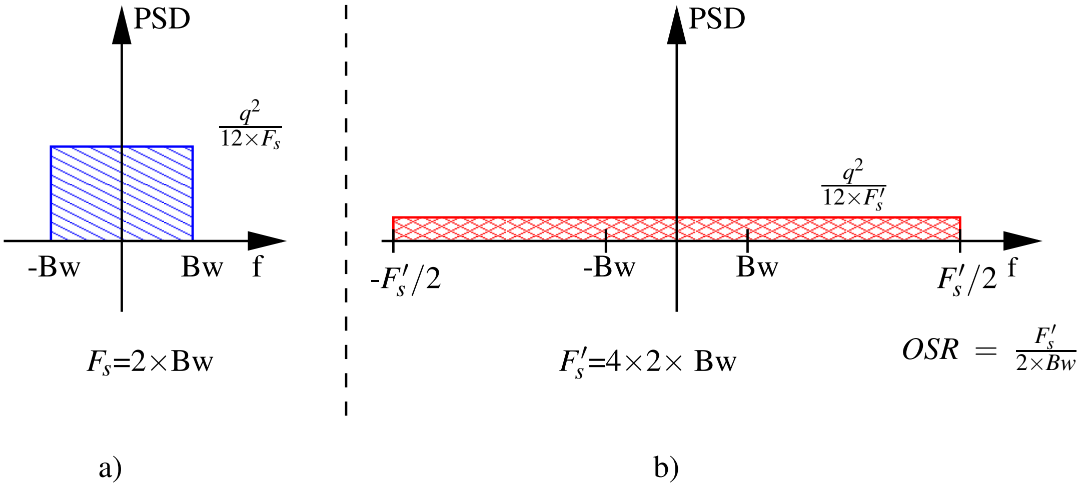
PSD
$\frac{q^2}{12\times F_s}$
-Bw
Bw
f
F_s=2×Bw
a)
PSD
$\frac{q^2}{12\times F_s'}$
$-F_s'/2$
-Bw
Bw
$F_s'/2$
f
F_s'=4×2× Bw
$OSR = \frac{F_s'}{2\times Bw}$
b)

FIGURE 3 – Power Spectral Density (PSD) a) without oversampling b) with an oversampling - OSR=4

Question 3.4 *Fix OSR to 50 in the file `flash.m` and nb to the value obtained in the previous question. Plot and analyze the output signal in time domain and in frequency domain also. Compare the practical SNR to the theoretical SNR.*

Despite the area reduction allowed thanks to the oversampling, the Flash architecture remains inadequate for an ADC with these specifications. Actually, Flash ADCs are rather used for the conversion of wide band signals (hundreds of MHz) with low resolution (4-6 bits) as shown in Fig. 4. For the wanted specifications, the Delta Sigma architecture seems to be the most suitable.

4 Delta Sigma ADCs - Time domain analysis

The operation of Delta Sigma ($\Delta\Sigma$) ADCs is based on two principles : oversampling and noise shaping (Pushing the quantization noise out of the band of interest). The block diagram of a first order $\Delta\Sigma$ modulator is shown in Fig. 5. It consists of an integrator followed by a low resolution flash quantizer (1 to 5 bits in practice) and a feedback loop that stabilizes the modulator. All these blocks operate at a speed $f_s = OSR \cdot 2 \cdot Bw$. The digital decimation filter at the output allows the transition from a high speed low resolution signal to a low speed high resolution signal. Let us begin by a time domain analysis of the $\Delta\Sigma$ modulator. The first test is carried out with a DC signal.

Question 4.1 *Load the file `SDt.m` into the editor. Set the modulator order to 1, the number of bits n to 1 and the input to 'DC'. Plot the input and the output signals $x[n]$ and $y[n]$ for the following input amplitudes AMP = - 0.5,0,0.5. Determine, for each of the amplitudes, the output sequence that is repeated*

Let us now perform the analysis with a sine input.

Question 4.2 *hange the input to 'AC'. Plot the input and the output signals $x[n]$ and $y[n]$ for the 3 following input amplitudes AMP= 0.2,0.5,0.8. Report the output variations around the maximums, minimums and zeros.*

Question 4.3 *Based on the results of the two last questions, describe the relationship between the output signal $y[n]$ and the input signal $x[n]$ in a $\Delta\Sigma$ modulator.*

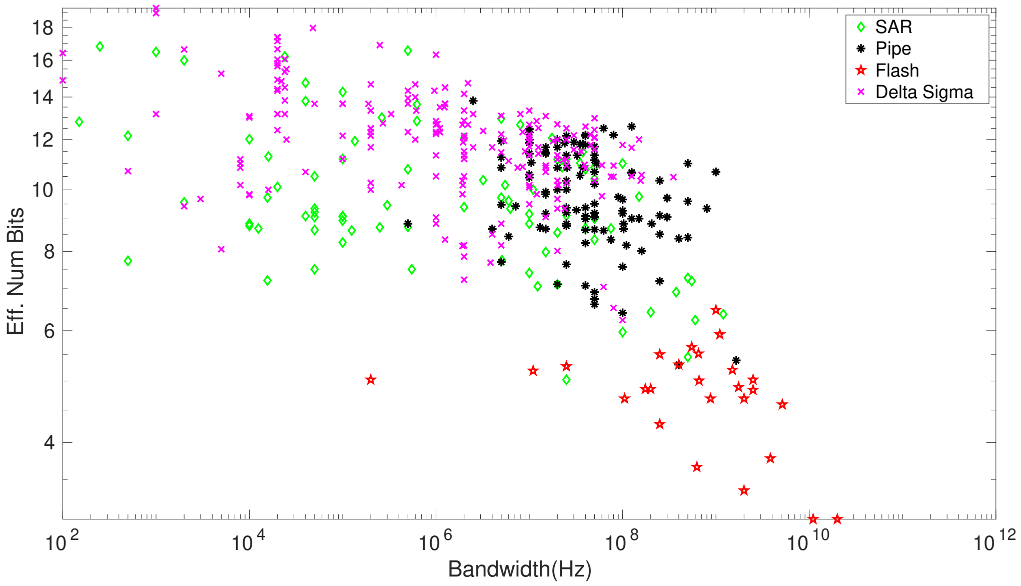

FIGURE 4 – ADC from the literature placed in the bandwidth - resolution space

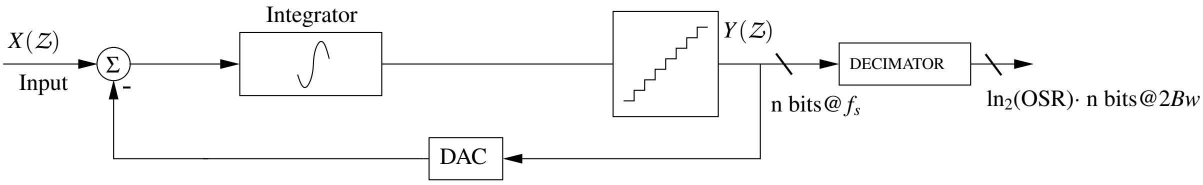

FIGURE 5 – First order $\Delta\Sigma$ modulator

5 Delta Sigma ADCs - Frequency domain analysis

Fig. 6 illustrates a possible mathematical modeling of a first order $\Delta\Sigma$ modulator. As can be seen, the integrator is replaced by its $\mathcal{Z}$ transform and the quantizer is replaced by a white noise source.

Question 5.1 *Using the mathematical model proposed in Fig. 6, express the output signal $Y(\mathcal{Z})$ as follows :*

$$Y(\mathcal{Z}) = STF(\mathcal{Z}) \cdot X(\mathcal{Z}) + NTF(\mathcal{Z}) \cdot N(\mathcal{Z}),$$

where $STF(\mathcal{Z})$ and $NTF(\mathcal{Z})$ are respectively the input signal transfer function and the quantization noise transfer function.

Question 5.2 *Replace $\mathcal{Z}$ by $e^{j\omega T_s}$ to compute $NTF(j\omega)$. Draw the shape of its modulus from 0 to $\frac{f_s}{2}$. Explain the interest of having this kind of shaping for the quantization noise ?*

Question 5.3 *Load the file `SDf.m` into the editor. Set the order of the modulator to 1 and the OSR to 50. Plot the output signal spectrum. Compare the shape of the quantization noise to the result obtained in the previous question. Read the value of the obtained SNR. Repeat the same test for OSR of 25 and 100. Discuss the obtained results.*

Fig. 7 shows the block diagram of a second order $\Delta\Sigma$ modulator. Its noise transfer function $NTF(\mathcal{Z})_{ord2}$ can be approximated to $NTF(\mathcal{Z})^2_{ord1}$.

Question 5.4 *In the file `SDf.m`, set the order of the modulator to 2. Report the SNR value for OSRs of 25, 50 and 100. What configuration would you choose to implement the ADC.*

6 Delta Sigma Modulator - Implementation in Cadence

In this section, we will focus on the implementation of the Delta Sigma on Cadence. We will use a switched capacitor differential implementation.

Question 6.1 *In Virtuoso, select Tools ⟶ library Manager and then go to the library IC_filter and open the cell DeltaSigma_modulator, view schematic.*

For the switched capacitor implementation, we need two complementary clocks S and T with a period of 1 ns.

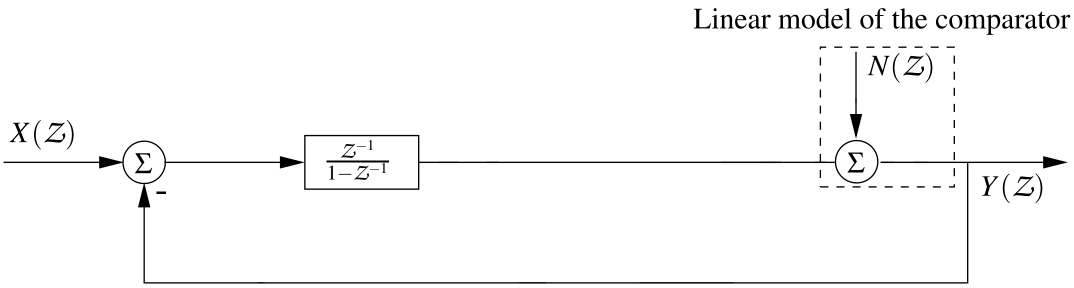
Linear model of the comparator
$N(\mathcal{Z})$
$X(\mathcal{Z})$
Σ
-
$\frac{\mathcal{Z}^{-1}}{1-\mathcal{Z}^{-1}}$
Σ
$Y(\mathcal{Z})$

Figure 6 – Mathematical model of a first order $\Delta\Sigma$ modulator

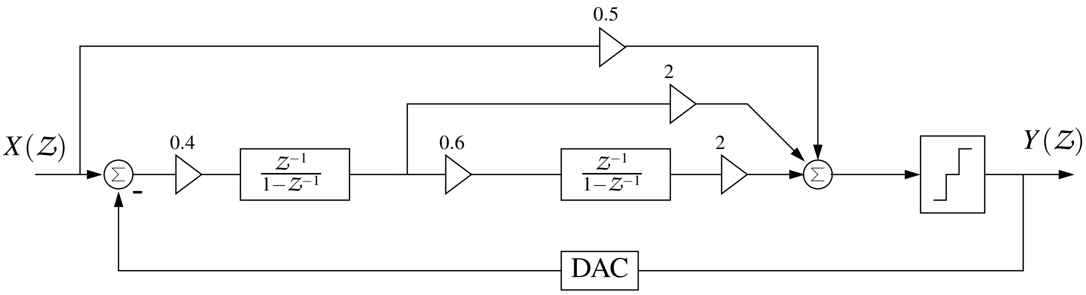

FIGURE 7 – Second order ΔΣ modulator

Question 6.2 *Instanciate two clock sources*[2]*. Adjust the S clock with following parameters : Source type :pulse ; Delay Time :0 ; Zero Value :0 ; One value :1 ; Period :1n ; Rise time :10p ; Fall time :10p ; Pulse width :480p*
Adjust the T clock with the same parameters but with a Delay Time of 500 ps in order to make a complementary clock.

Question 6.3 *Build your modulator by instantiating and connecting your components*[3]*.*

Question 6.4 *Set the values*[4] *of your components to the values of figure 7*

Question 6.5 *Name*[5] *the differential input INP and INN and the DAC differential output DACP and DACN.*

Now, that our circuit is built, it is time to simulate it. As our circuit is a switched capacitor circuit, using AC analysis is not possible. We will hence only carry out transient simulations. It is worth mentionning that it is possible to run small signal analysis on switched circuits but this requires using periodic steady state (PSS) and Periodic AC (PAC) simulations.

Question 6.6 *In the Library Manager, library IC_filter, cell DeltaSigma_modulator, open the spectreTran view. This will open the simulation environment. Push on the play icon to launch the simulation. Observe the results and compare them to the system level simulations performed with Matlab/Octave.*

[2] To instantiate a component, push i. A window will open to allow you to select the symbol view of your component. vsource and ground (gnd) are in library analogLib. For integrators, adder and comparator, use the following cells respectively Integrator, Adder and ComparatorDAC of IC_filter.
[3] To connect the different components together, use w
[4] Select the component and push q
[5] To name a net, push l, write the name and then select the considered net