LUND UNIVERSITY

Electrical and Information Technology

Integrated A/D and D/A Converters (ETIN55)

Lab1 Manual

Pietro Andreani
Martin Anderson

October 2021

Lab 1 – Fundamentals of A/D conversion

This lab will introduce the fundamental properties of A/D converters, which will be studied in both time and frequency domain. Furthermore, non-ideal properties of samplers and quantizers will be studied as well.

The lab deals with the following issues:

1. Discrete Fourier Transform and its FFT implementation
2. Fundamentals of sampling (bandwidth limitation of A/D conversion)
3. Non-ideal behavior of samplers (circuit noise, nonlinearity and clock jitter)
4. Fundamentals of quantization; noise limitation in A/D converters; statistic properties and power of the quantization error; effect of applying dither.
5. Non-ideal behavior of quantizers (circuit noise, nonlinearity)
6. Performance budget for an ADC. Calculation and system simulation of total noise in an ADC

MATLAB package

The study the fundamental behavior of an ADC, a Matlab package will be used, contained in the file **MatlabLab.zip**. It includes functions for signal generation, sampling and quantization, as well as for performance evaluation. All Matlab functions have extensive help texts that can be accessed by typing

```
>> help filename
```

The structure of the Matlab package allows the study of sampling and quantization separately, but also their combined effect (Figure 1).

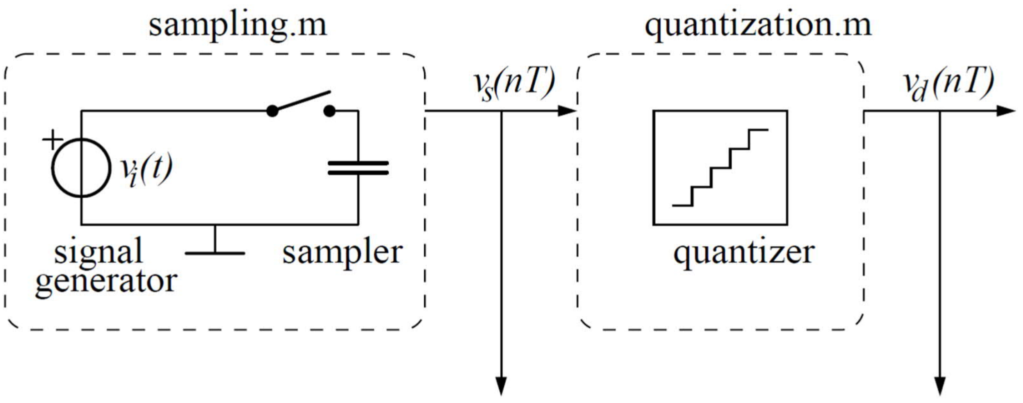

Figure 1. Illustration of the intended usage of the provided MATLAB files.

It contains the following files:

- **sampling.m** implements a signal generator and a simple sampling circuit in one file for best accuracy.
- **quantization.m** implements a quantizer that rounds its input signal to the closest lower level.
- **adcfft.m** performs an FFT with or without windowing and calculates an output spectrum.
- **adcperf.m** analyses the spectrum from adcfft.m to find the noise floor, harmonics and other distortion. It calculates SNR, SNDR, THD and SDR.
- **ADCdemo.m** demonstrates how the other files can be used by modeling several non-ideal effects at the same time.
- **hann1.m** and **hann2.m** implement first and second order hann windows, respectively.
- **INLDNL.m** is a script for calculating INLs and DNLs, used in the spectreVerilog lab.

Homework exercises

1. Study this lab carefully.
2. Read the theory on data converter fundamentals [DataConverters, Maloberti], pp. 1-75.
3. Download the zip file `MatlabLab.zip` from the course homepage, and unzip it in your Matlab directory:
   ```
   > unzip MatlabLab.zip
   ```
4. Investigate the different Matlab files used in this lab. Especially try to run ADCdemo.m and understand the code.
5. Define and explain the following performance measures for ADCs: SQNR, SNR, SNDR, THD, HD_2, HD_3, and IM_3.
6. What is aliasing? What is the maximum bandwidth the input signal of a sampler can have, in order to avoid aliasing?
7. Give an expression for the quantization noise power for a quantizer with R bit resolution.
8. Assuming a single full swing sinusoidal input signal with amplitude Ai, what is the theoretical maximum signal-to quantization noise ratio (SQNR) of an R bit ADC?
9. Assuming you have an additive white Gaussian noise in the quantizer besides the quantization noise. How large should the power of this noise be, in order to limit the degradation of the system SNDR to 3 dB?
10. Assuming a 100 MHz input signal, how much clock jitter (the standard deviation in ps) can be tolerated in order to make the power of this error smaller than half the quantization noise power of a 10 bit ADC?
11. Try to finish as many of the laboratory exercises as possible (on your own!) before the scheduled lab event.

Laboratory exercises

Part 1: A study of the DFT

In this part, we study some properties of the DFT (FFT). You shall write your own Matlab code in the file *MySolutions.m*. Look at the code in *ADCdemo.m* to understand how to call the functions *sampling.m, adcfft.m* and *adcperf.m.*

If nothing different is stated, use the following data in this part:

- A_{in} = 1V
- f_{in} = 9.97 MHz
- f_s = 81.92 MHz
- N = 8192
- # of averages = 16

1. To start, we shall try coherent sampling, where the observation time (N/f_s) is an integer multiple of the signal period ($1/f_{in}$). This means that $N/f_s = m/f_{in}$,where m is an integer. According to theory [Maloberti, p. 28], a rectangular window will give a nice spectrum. If the signal amplitude is 1V, the input signal frequency 9.97 MHz and the sampling frequency 81.92 MHz, and we take a 8192-point FFT, what m-value is used? Take 16 averages. Plot the spectrum with the y-axis on a log scale. At what level is the signal peak? Over how many frequency bins is the signal power found? What is the cause of the "noise floor" (notice that it is exceedingly low)?

   Question 1.1: Where does the "noise" in your plot comes from?

2. Try the same signal and FFT, but change window to a hann1 window.

   Question 1.2: What is the level of the signal peak? Over how many frequency bins is the signal power found? How far below the signal peak are the two neighboring bins?

3. Now consider the case when the observation time is not an integer multiple of the signal period. Try input signal frequency 9.97 MHz, sampling frequency 80MHz and an 8192 point FFT with rectangular window. Plot the spectrum.

   Question 1.3: What happens compared to experiment 1.1? Why?

4. Redo the experiment 1.3 with a hann1 window.

   Question 1.4: Was the noise levels improved compared to experiment 1.3? Why? What about a hann2 window? Explain.

   Question 1.5: When should windowing be used? Are there cases when a rectangular window is better?

5. Set f_s back to 81.92MHz. After generating and sampling your input signal, add a white random noise with *rms* level corresponding to 10-bit quantization noise by using the MATLAB function *randn*(). Take into account that *randn* returns normally-distributed random numbers with zero mean and unity variance. Make an FFT where you use 64 averages, and another FFT with no averaging. Observe and explain the difference in the frequency spectrum.

   Question 1.6: When and with what kind of signals should averaging be used?

   Question 1.7: Describe with your own words, how to take a good FFT for ADC evaluation.

   Question 1.8: Using *adcperf.m,* find the SNR and the SNDR of the signal in the previous exercise.

Part 2: A study of sampling

If nothing different is stated, use the following data in this part:

- A_{in} = 1V
- f_{in} = 9.97 MHz
- f_s = 81.92 MHz
- N = 8192
- # of averages = 16
- window = hann1

6. Use *sampling.m* to add a third order harmonic to the signal in the previous exercise. For example, set the **k3** parameter to 0.01. Run an FFT of an input signal of 9.97 MHz and one for an input signal to 39.19MHz.

   Question 1.9: At what frequency can you observe signals after the sampling process? What has happened?

   Question 1.10: What is the fundamental bandwidth limitation of an A/D converter?

   Question 1.11: Explain with your own words what an anti-alias filter is, and why it is necessary in most applications.

7. Write a script where you study the noise from a simple sampler using *sampling.m*. Include the parameter **Cs** in the call to *sampling.m*, indicating that the **kT/C** noise should be generated. Vary the size of **Cs** from 0.01 pF to 10 pF in linear steps of 0.2 pF. Plot the simulated SNR vs. Cs. Calculate the theoretical SNRs for an 8, 10, 12 and 14-bit ADC and indicate those theoretical levels in the plot with straight lines. Furthermore, calculate and plot the theoretical SNR curve due to the kT/C noise.

8. Write a script to study the effect of clock jitter. Add a random clock jitter using *sampling.m*. Vary the input signal frequency. Plot the resulting SNDR versus the input signal frequency for three different clock jitter standard deviations σ = 1ps, σ = 5ps, and σ = 15ps. Calculate the theoretical limitation for the above jitters as a function of input signal frequency and include the result in your plot.

9. Approximate a 100 MHz input signal using f_{in} = (3277/8192) f_s (which ensures coherent sampling), and use a 250 MHz sampling frequency. Sweep the standard deviation for the clock jitter from 1fs to 20ps in steps of 1ps and plot the simulated SNDR limitation. Check that if f_{in} = 100 MHz (i.e., there is no coherent sampling), the SNDR is dominated by the spectral leakage.

   Question 1.12: How well does the simulation result of the last exercise correspond to your calculation in the homework exercise?

Part 3: A study of quantization

If nothing different is stated, use the following data in this part:

- A_{in} = 1V
- f_{in} = 9.97 MHz
- f_s = 81.92 MHz
- N = 8192
- # of averages = 16
- window = hann1

10. The assumption that the quantization noise can be modeled in frequency as having a rectangular power spectral density (from DC to Nyquist) is frequently used. You shall investigate if this model is accurate in this and the next experiment. Use again *randn*() to add a Gaussian distributed noise with the same power as an 8-bit quantization error to a sinusoidal signal. Run the FFT, plot the spectrum and calculate the SNDR using *adcperf.m*.

11. Remove the Gaussian noise, keep the same input signal, and quantize it using the file *quantization.m* instead. Run the FFT, plot the spectrum and calculate the SNDR.

    Question 1.13: What happens to the random noise floor and to the quantization noise floor if you change the number of averages? Try # of averages=1. Does averaging have the same effect on random noise vs quantization noise? Explain.

12. Let us further investigate the quantization error. Generate a ramp starting at -1V and ending at 1V, and use *quantization.m* to quantize it with 3-bit resolution. The time vector can be created as $t = (1/f_s)(0 : len - 1)$. Plot both ramp and quantization error. Generate now a sinusoidal input, with f_s = 81.92 MHz, f_{in} = 9.97 MHz, # of averages = 2, $len = N\cdot$(# of averages). Plot two periods of the input signal and the corresponding quantization error.

13. **Question 1.14**: Is quantization a random process? Is the quantization error a random signal?

    Question 1.15: Is the signal power of the quantization error the same for your calculated rectangular distributed noise as for the real quantization process?

14. Add a Gaussian noise source with the same power of an R+1 bit quantization noise to your sinusoidal input signal in the second part of experiment 12 above (such a noise signal is often referred to as dither, and is sometimes used in Delta-Sigma modulators). Set R=8. Since you have additive noise, you may have to lower your input signal amplitude by some mV to avoid clipping in the quantizer. Run the FFT, plot the spectrum and calculate the SNDR.

    Question 1.16: What effects does dither have on the output spectrum? On the SNDR?

15. Redo the last experiment with a 1.01 V input signal. Calculate an FFT, plot the spectrum and calculate the SNDR.

    Question 1.17: What happens to the performance when the quantizer is clipping?

Part 4: ADC system noise – calculation and simulation

16. Obviously, the quantization fundamentally limits the accuracy of the A/D conversion. However, there are also nonlinearities, circuit noise and digital crosstalk noise in the implementation of an ADC. Assume all error sources are uncorrelated, and that they can be referred to the input of the ADC. Then the total noise is the sum of the input-referred circuit noise and the quantization noise. To investigate the performance of an R-bit quantizer with additive input-referred noise, do the following: take f_s = 81.92 MHz and f_{in} = 9.97 MHz (default values) and add Gaussian noise corresponding to a 10-bit quantization error. Add a second Gaussian noise source and sweep its power from 0 to six times the quantization noise power.

    Question 1.18: How large *rms* noise (in LSBs) gives a -3 dB performance degradation in SNDR? A -6 dB degradation?

17. The *ADCdemo.m* file contains an ADC system model. You will perform simulations on the ADC system model by running *ADCdemo.m*. The included ideal ADC parameters and non-ideal effects are listed below:

    i) *rms* value of the clock jitter: 8ps
    ii) sampling capacitor: 450fF
    iii) sampling nonlinearity: k_2 = 0.004, k_3 = 0.007, k_4 = 0.0025, k_5 = 0.004
    iv) resolution: 12 bits
    v) sampling frequency: 250 MHz
    vi) max. input amplitude = 1 V
    vii) *rms* value of the thermal noise of the quantizer = $(\Delta/4)/\sqrt{12}$, where Δ is the quantization error

    1. Make a simulation with a sine wave input signal, approximately at $f_{in} = f_s/10$. Calculate an FFT and take a careful look at the spectrum.
    2. Sweep the input signal amplitude from 0.001 mV to 1.42 V. Plot the SNDR, SNR and SFDR as a function of input signal amplitude.

    Question 1.19: What is the reason for the performance degradation at high input signal levels? What limits the SNR and SNDR respectively?

    3. Sweep the input signal frequency from close to DC to $f_s/2$. Plot the SNDR, SNR and SFDR as a function of input signal frequency.

    Question 1.20: What is the reason for the performance degradation at high input signal frequencies?

    4. Add a few lines of code to *ADCdemo.m* to plot ENOB versus input frequency, and state the ERBW.