DAT116 (Mixed-signal system design)
Lab 3: Quantization and jitter

Lars Svensson
`larssv@chalmers.se`

Version 1.7, November 26, 2018

1 Introduction

This lab session will be carried out in MATLAB and Simulink, so will not require Cadence access. Tasks are described somewhat briefly; you will need to use your experiences from previous labs to carry them out.

2 Preparation

In the MATLAB documentation, find the entries for the `1-D Lookup Table` block and consider how you might specify its index and data arrays in order to model a quantizer.

A Simulink simulation can be launched from MATLAB code with the command `sim`. Study the documentation for this command.

Make sure you know how `for` loops work in MATLAB.

3 Uniform quantization

In this first experiment, we will use pure sine waves as input for an A/D converter. We will model the converter in Simulink with a `1-D Lookup Table` block. Data to fill the lookup table will be constructed in MATLAB. Two functions, `lutix` and `lutdata`, are available to help with this task.

- Launch MATLAB; set up the current directory and the load paths in a suitable way; and launch Simulink.
- Open a new simulation model with the name `lab3a`. Insert a `Sine Wave` source, a `1-D Lookup Table` block, and a `To Workspace` block. Connect the input of the `1-D Lookup Table` to the `Sine Wave` and its output to the `To Workspace` block.
- Modify the parameters of the model and of the `Sine Wave` block to use 1024 samples and to fit 5 cycles into your window. Set the sine wave amplitude to 1, which will correspond to the full range of the quantizer. Select a fixed-step solver when you specify the sample count.
- Choose a MATLAB variable name in the `To Workspace` block. Remember to select the save format `Structure with Time`.

The lookup table will be used to model a quantizer. We will use `lutix` and `lutdata` to generate the data for the table.

- Make sure that the `lutix` and `lutdata` functions are available in a directory on your `path`.
- In the MATLAB command window, give the commands `lutix(4)` and `lutdata(4)`. Study the outputs.
- Next, plot the output of `lutdata(15)` with respect to `lutix(15)`, and save the figure for your report. Play with the arguments for the two functions, always using the same value for both. How does the number of levels in the `lutdata` array correspond to the argument value?
- Open the parameter dialog of the `1-D Lookup Table` block. Two parameters, `Table data` and `Breakpoints 1`, are used to specify the table contents. Enter `lutdata(4)` in the first of these fields and `lutix(4)` in the other one. With these parameters, how will the lookup table map input values to output values?
- Run the simulation and view the signals before and after the quantizer. Make a plot of the quantizer output for your lab report.
- Use `sigspectrum` with the quantized signal as an argument. Does the spectrum correspond to your expectations? Generate a plot for your lab report.

You will now calculate the SNR values for uniform quantizations at varying resolutions. You will use `sigspectrum`'s ability to extract the power value vectors for each frequency. You will generate one SNR value per Simulink run. You will need to be able to make a plot of your results and include that plot in your lab report.

Recall that Simulink block parameter values can be MATLAB expressions, which may include values of variables from the MATLAB workspace. This feature makes it possible to use the resolution in bits as a loop iterator, and calculate the lookup-table contents in each iteration using the `lutix` and `lutdata` functions with the number of quantization levels as an argument.

- The argument of the `lutix` and `lutdata` functions specifies the number of quantization levels of the converter. How does the number of *levels* correspond to the number of *bits* of resolution?
- Use MATLAB's programming features to execute the `lab3a` model for each number of *bits* from 3 to 14. For each parameter value, calculate the conversion SNR. Plot the SNR (in decibels) as a function of the resolution and save the plot for your report.

What were your expectations, and does the plot correspond to them?

4 SNR vs input power

Above, you used MATLAB code and the `sim` command to run a set of simulations with different parameters. The same technique can be used for other parameter sweeps as well. Your next task is to sweep the input power of a quantizer and record the SNR values.

- Make a copy of `lab3a` and name it `lab3b`.
- In `lab3b`, set the parameters of the `1-D Lookup Table` block to represent a resolution of 10 bits.
- Set the amplitude parameter of the `Sine Wave` block to the value of a MATLAB variable, for example one named `foo`.

You will refer the input signal level to the level that corresponds to a full-range sinewave signal. What is the power of a sine wave that covers the full range of ± 1, that is, a sine wave of amplitude 1?

- Use a `for` loop as in the previous section to simulate the system with input powers from −20 dB to +3 dB relative to the full-range signal, in steps of 0.5 dB. For each value, you will need to calculate the `foo` value which corresponds to the input power. Calculate the SNR values in dB as before, and plot the SNR as a function of the input power (also in dB). Make a hardcopy for your lab report.

5 Non-uniform quantization

Above, we have used a lookup table to model a quantizer, that is, an A/D converter. In fact, it is even more natural to use such tables for modeling D/A converters, since in that case, input values are restricted to a limited set. Thus, we can use a simpler table than in the A/D case; but the D/A model will not be a perfect mirror image of the A/D model.

We will use signed integers as indices into the lookup table. For an N-bit resolution, a suitable value for `Breakpoints 1` would be given by the expression `-2^(N-1):(2^(N-1))-1` (for a sanity check, evaluate this expression in MATLAB with some small values of `N`!).

- Make a copy of the model `lab3b` and call it `lab3c`.
- Open the `1-D Lookup Table` block and replace the value of `Breakpoints 1` with values suitable for simulating a 10-bit D/A converter.

We will investigate the R2R ladder converter, described in Maloberti 3.3.8. The function `r2r` is available; it returns a voltage vector containing the output values generated by an R2R ladder. The desired resolution, *in bits*, is given as a parameter to `r2r`.

- In the MATLAB command window, try out `r2r` a few times with small integers as parameters.
- Replace the `Table Data` parameter value in the `1-D Lookup Table` with a suitable call to `r2r`.

The lookup table expects integer input data covering a certain range. The sinewave data must be converted correspondingly.

- Connect a `Gain` block and a `Data Type Conversion` block between the sinewave source and the lookup table input.

- Open the `Data Type Conversion` block and select the `Output data type` to `int32`. Also, select `Real World Value (RWV)` and `Nearest`, respectively, in the other selectors.

By default, the `1-D Lookup Table` block uses the same data type for the output as is used on its input. Here, we need a floating-point output for an integer input. It is also necessary to specify how the lookup table is to interpolate between values[1].

- Open the `1-D Lookup Table` block. Select the `Data Types` pane and set the output data type to `double`.
- Next, select the `Algorithm` pane, and set the `Interpolation method` to `Flat`. Click `OK` to accept the block parameter values and close the dialog.
- Open the `Gain` block and set the gain factor to expand `sin(x)` to use as much as possible of the input value range without overflowing. Accept and close.
- Run one simulation and verify (with `sigview`) that the output values from the lookup table represent a sine wave.

With accurate resistor values, an R2R ladder will yield accurate signal values for each input value. In practice, resistor values are never perfectly accurate. In order to model this phenomenon, the `r2r` function takes an optional second parameter, `sigma`, which specifies the relative standard deviation of the values of all resistors (so a value of 0.1 represents a standard deviation of 10% of the actual resistor value).

We now have all necessary components to investigate how the SNR depends on the signal level for an R2R-based D/A converter with non-ideal resistors. There is one caveat: `r2r` uses a pseudo-random function to introduce resistance inaccuracies; and Simulink evaluates parameter expressions before each simulation run. Thus, if `r2r` is called directly in the `Table Data` field, a different vector will be used in each simulation. Since we wish to investigate how a single converter handles many different input levels, we need to execute `r2r` in MATLAB, save the resulting vector in a MATLAB variable, and use the variable name in the `Table Data` field.

- Generate a 10-bit-converter value vector using `r2r` with a second argument specifying a resistor-value standard deviation of 2%. Save the vector as a MATLAB variable.

[1]We will in fact not ask the block to interpolate, but the block does not know that...

- Use the MATLAB variable name as the `Table Data` parameter value in the lookup table.
- Setup a simulation sweep as you did for `lab3b` and collect the SNR values. Produce a diagram of SNR vs signal level. Make a hardcopy and save for your report.
- Repeat the previous three points (generate value vector, use as `Table Data`, simulate and plot) but this time with a standard deviation of 4% instead. Compare with the previous result. Did the outcome agree with your expectations?

Note: the absolute SNR values depend on the specific set of vector values produced by the call to `r2r`. As these values depend on random resistor value deviations, you should not expect to get exactly the same results if you re-run this experiment.

- Re-generate a new value vector with 4% resistor standard deviation, re-run the simulations, and plot the SNR values as before. Save the plot for your report. Discuss the differences of the two nominally identical runs.

6 Sample jitter

It is not immediately obvious how to simulate sample jitter with the simple Simulink setup we use in these labs; after all, the simulations use a fixed time step and sampling is therefore inherently uniform. Here, we will use the time-derivative of the sampled signal explicitly[2].

- Open a new Simulink model and name it `lab3d`.
- Again, use a single `Sine Wave` block to generate the input signal, fitting 5 cycles in 1024 time points, as before. Set the sine wave amplitude to 1. Use a `Derivative` block to calculate the slope of the input signal; multiply the slope (using a `Product` block) with the output of a `Uniform Random Number` block; and add the resulting product to the signal before saving it to the MATLAB workspace.

The `Uniform Random Number` block lets you specify upper and lower limits for the random numbers it generates. We will use symmetric limits here.

[2]F. Maloberti et al. Behavioral modeling and simulations of data converters. In *Proceedings of XVI IMEKO World Congress*, Vienna, Austria, Sep 25–28, 2000.

- Set the parameters for the random number limits to achieve a maximum deviation of 10% of the sample interval. Also set the sample time to 0 in order to make the `Uniform Random Number` block use the same sample time as the rest of the simulation.
- Run your simulation. Use `sigview` and `sigspectrum` to investigate the resulting signal. (You may have to zoom in on the signal to see the irregularities; it may also be good to call `sigview` with both the original and the jittered signal as arguments.) Observe the level of the noise floor in the spectrum plot.
- Save a copy of the signal you just generated in a new MATLAB variable. Increase the frequency of your `Sine Wave` signal by a factor of 10, and re-run the simulation. View the spectra of the old and the new signals in the same `sigspectrum` window. Does the result correspond to expectation? Save the plot for your report.
- Extract the power spectrum vector values and calculate the resulting SNR for the two cases above. Does the difference meet your expectations?

As a final task, you will sweep the input level for the jitter-affected sample setup and record the signal and noise powers as functions of the input power.

- Set up and run a set of simulations for input power levels of -40 dB to $+3$ dB in 1-dB steps, using the `for` loop method of the previous sections, and the higher input frequency from the previous task. Create and save a plot of the signal and noise powers as functions of the input power, using all dB scales. Does the plot correspond to expectations? What would an SNR plot look like?

7 Wrap-up

After completing this lab, you are supposed to be able to carry out the following tasks:

- Set up simulations of uniform and non-uniform quantizers in Simulink.
- Use MATLAB to sweep a variable and carry out Simulink simulations for each of the variable values.
- Use Simulink to estimate the influence of sample-clock jitter on the SNR of a sampled-signal system.

Reflection questions:

- We know that two uncorrelated signals of equal power will, when added, combine to a power 3 dB larger than that of either signal. Thus, if you have an overall SNR requirement of (say) 50 dB and two uncorrelated errors (such as a quantization error and a jitter error) of *equal* power, then *each* of those errors needs to be 53 dB below the signal for the overall requirement to be met. If however the quantization error is somewhat larger, just 52 dB below the signal, it is still possible to meet the overall requirement if the jitter error is reduced accordingly. To what level (in dB) will it need to be reduced?

- The `r2r` MATLAB function assumes that all ladder-resistor values have the same standard deviation. As you have seen previously, the resistance value of an on-chip resistor depends on its aspect ratio L/W; and it is possible to improve the matching of on-chip resistors by spending more silicon area, that is, by increasing both L and W by the same factor.

  For a given *total* resistor area, how would you distribute that area along the rungs of the R2R resistor ladder to reduce the errors? (A qualitative discussion is sufficient; disregard any influence on dynamic behavior.)