Experiment 1: Sampling and Quantization

Bruno Korst - bkf@comm.utoronto.ca

Abstract
In this experiment, you will see the effects caused by changes in sampling rate and quantization levels on a digitized (i.e., sampled) sinusoidal signal. You will start by observing aliasing. On the second part of the experiment, you will move on to changing the number of bits assigned to each sample of the input signal, and will observe the presence of quantization noise.

Keywords
sampling — sampling rate — aliasing — quantization levels — quantization noise

Contents

Introduction

The purpose of this experiment is to introduce *sampling* and *quantization*. It is fair to say that every textbook on Digital Signal Processing (DSP) starts by describing how an analog signal is converted into a digital signal prior to being processed [1], [2]. It is done by taking samples of the signal and assigning numbers to these samples. There are limits to how frequently you should take samples of a signal, and to how many bits you should assign for every sample. In this experiment, you will play with these limits. It is a good idea if you review your notes on aliasing and downsampling from introduction to signals and systems, and bitwise operations from microprocessor theory.

This experiment will be divided into the following parts:

- First, you will run a simulation model to create *aliasing*;
- Second, you will verify the results of this simulation on the DSP target hardware.
- Finally, you will force a change on the number of quantization levels utilized to convert the input signal. This will be done by applying a *bit mask* on the incoming samples, leading to a change in *quantization error*.

Note that the lab **answer sheet** is to be done in groups of two students. You should print the answer sheet to bring to the lab. In the lab you will follow this procedure online and record your answers in the answer sheet.

1. Experiment

1.1 Sampling

In this section you will modify the sampling rate, or sampling frequency, used to sample a signal. The objective is to verify the effect of *aliasing*. You will simulate it first, and then will implement it on the DSP platform. Since the DSP platform already

offers an audio CODEC sampling the input at 48KHz, we will use *decimation* to achieve a different sampling rate and observe aliasing.

1.1.1 Simulation

Let us build first a system that works properly and then break it. In `Simulink`, this will be done with the use of the **downsample** block, found in the `Simulink` library. Downsampling is also known as **decimation**. The most common representation of decimation in block diagrams is a square with an arrow pointing down beside a number that is the **decimation factor**. As the name implies, when you decimate or downsample a discrete signal by a factor, you *reduce* the sampling rate of the signal by that factor. This is to say that effectively you will discard a number of samples when you decimate a sampled signal. For instance, if you have a 1KHz signal sampled at 48KHz, you will have 48 samples per period. When you decimate this signal by a factor of 2, you will discard every other sample and end up with 24 samples per period. This is equivalent to sampling the 1KHz signal at a 24KHz sampling rate.

Now open a new model on `Simulink` with a DSP Sine wave generator attached to a time scope and a frequency domain scope. You can build your own frequency domain scope by using a `Buffer` block, a `Magnitude FFT` block and a `Vector Scope` block. Set the buffer to 1024. Double click on the `Magnitude FFT` and select `Magnitude`, as opposed to the default. Double click on the `Vector Scope` and set the `Frequency Range` under the `Axis Properties` to `[0...Fs]`. Now adjust your sine wave to be $2V_{pp}$ and 22KHz, using a 48KHz sampling rate. You may want to try 21KHz as well and as an exercise try to recall how to explain the difference in amplitude. Make sure that the 22KHz runs smoothly. Now place the decimation block between your sine wave generator and the scopes. Set the decimation factor to be 2 and select "allow multirate".

If you select "force single rate", you are effectively selecting one sample and holding its value for the duration of the other (it's a factor of 2). This corresponds to applying a rectangular window [1 1] to a sample (i.e., hold one, discard the other), which will have the effect of a low pass filter in the frequency domain. You *do not want that*.

Your model should look like Figure 1 below. Run it and answer the questions in the answer book.

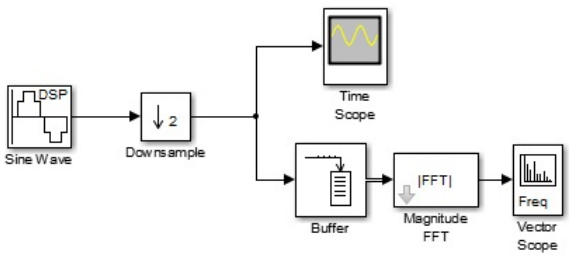


Figure 1. Suggested Downsampling Simulation

1.1.2 Implementation

Now you will observe the same effect on a real signal. Open `Code Composer Studio` and select the project named `ECE431_Exp01a_Sampling_Aliasing` as your active project. Compile, download and run the project. When you turn on the oscilloscope, put it in `Math Mode`, by pressing the big fat red button located right in the middle of the scope. You will only look at the frequency domain plot for this part of the experiment, as presented on Figure 2. It must be pointed out that the magnitude of the components decrease due to the attenuation of a low pass filter on the board.

For the input to your target hardware, set your signal generator to a $1.8V_{pp}$ sinusoid, with 1KHz frequency. One channel should output a decimated version of the input, and the other channel should output an unmodified sinusoid. This program decimates the input by a factor of 4, so your new sampling rate is 12KHz.

1.2 Quantization

This part of the experiment will be done without simulation. Stop the previous program and select now as active project the one named `ECE431_Exp01b_Quantization`. After doing so, build it and run it.

This program will leave the sampling rate untouched, but will allow you to modify the number of bits used to represent the incoming signal. The quatization done by the existing CODEC on the board provides 16 bits per sample, which means 2^{16} quantization levels. That is a very good resolution for an audio signal, for instance. The hardware you use determines the

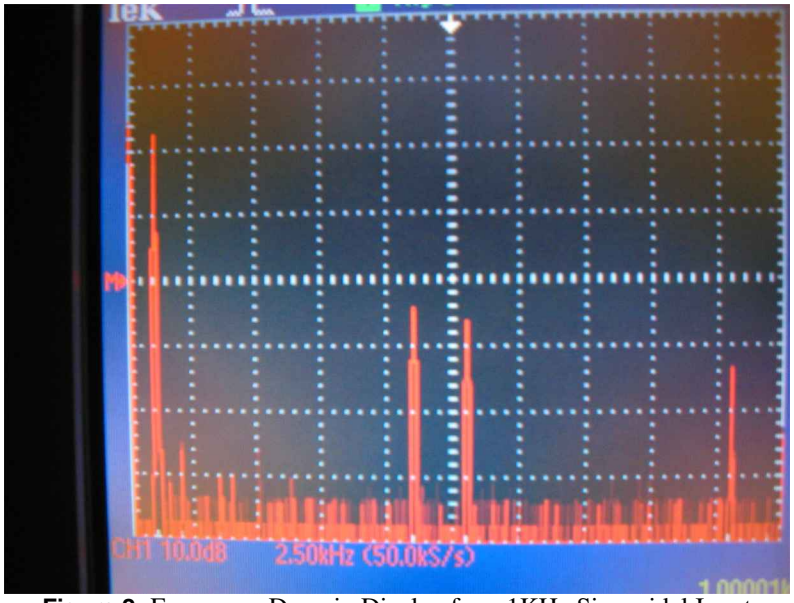


Figure 2. Frequency Domain Display for a 1KHz Sinusoidal Input

word size in which samples are represented, and for this case it is two channels with 16 bits per sample each (i.e., it is a 32-bit CODEC).

The experiment now is to limit the number of bits from the original 16-bit quantization, and observe the results on the oscilloscope. You will effectively increase the *quantization error*. As you have already reviewed, when you reduce the number of bits on the quantization, you increase the error introduced in the process by reducing the number of quantization levels available. The question to be asked is: how much error can you tolerate for your application? Or, in other words, how many bits do you really need in order to represent your signal and maintain the desired quality? Figure 3 below shows a sinusoid, its 3-bit quantization version and the resulting error.

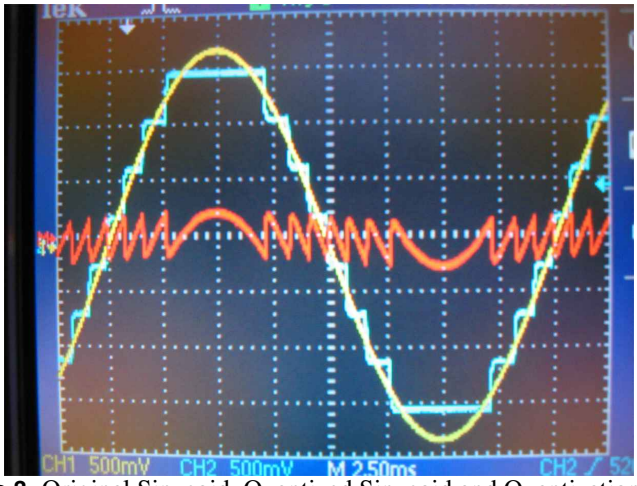
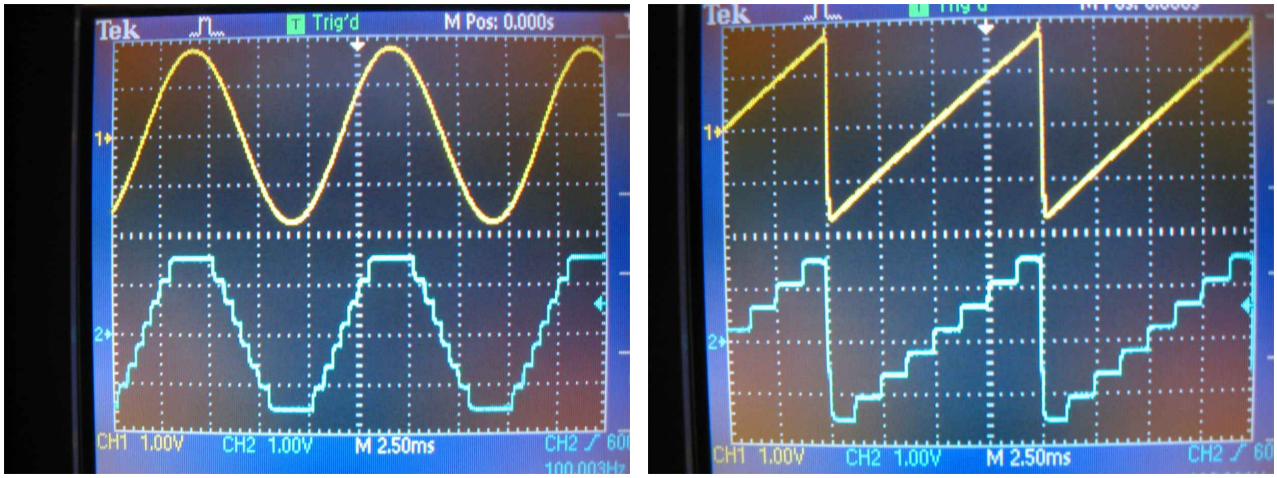


Figure 3. Original Sinusoid, Quantized Sinusoid and Quantization Error

You will utilize a "bit mask" to modify the word length of the incoming signal. You have likely utilized masking in a Microprocessors course. It consists of performing a logical AND operation between the data word passed on to the DAC and a "mask" that you select. Figure 4 below shows a sinusoid and a ramp, quantized with 3 bits, for instance. Notice the 8 levels of the output.

You will need to rebuild the project after every change you make on the code. Use an input signal of $1.8V_{pp}$ and 100Hz to answer the questions below. The reason why you are using $1.8V_{pp}$ is that you want to use the full voltage range of the CODEC, so that all bits of the word will be used. If you do not have the full range of the signal going into the CODEC, the different



(a) Sinusoid (b) Ramp (Sawtooth)

Figure 4. A Quantized Sinusoid and a Quantized Ramp

masks may not give you the right results.

2. Accomplishments

In this experiment, you acquired a better understanding of sampling and quantization by further probing into the limitations of the sampling frequency and the number of bits per sample. You verified the introduction of aliasing by sampling a signal at a rate lower than the one dictated by the sampling theorem. You also verified the consequence of reducing the quantization levels by limiting the number of bits utilized to represent the samples of a signal.

Acknowledgments

Thanks for all the students who have provided input on the previous versions of this experiment.

References

[1] Schafer R. W. Oppenheim A. V. *Discrete-Time Signal Processing, 3rd. Ed.* 2009.

[2] Manolakis D. K. Proakis J. G. *Digital Signal Processing, Principles, Algorithms and Applications, 4th. Ed.* 2006.