Introduction

This article will demonstrate how the Viterbi algorithm can be used to produce a pseudo-optimal discrete-time Sigma-Delta data converter.

Recap

In our previous installment, we used a national/continental sales tour to show how the Viterbi algorithm worked. If you have not already, you should go back and visit that post. Some of you may want to revisit it to recall the terminology and nomenclature.

A brief recap is that the Viterbi algorithm is useful when:

1. We are trying to find a "transmitted" sequence…
2. …that minimizes some error…
3. …between the sequence and some other "received" sequence.
4. Each point (called a symbol) in the sequence is picked from a finite alphabet (i.e. we are performing a minimization over a discrete set).
5. The error is a linear function of the hypothesized sequences.
6. The error function has finite memory (K past symbols).

Note: the terms "transmitted" and "received" are used when the Viterbi algorithm is being applied to a communication system. This application is what most people are familiar with, so I will borrow these terms here. The upshot is that the Viterbi algorithm will yield the optimal solution by inspecting each received sample r[k] and eventually finding the transmitted set of samples b[k] that minimizes $|b[k]*h[k] - r[k]|^2$. The * operator represents convolution, and h[k] is the error function's impulse response.

Sigma Delta

The sigma delta has a very similar objective: it is trying to minimize some in-band energy between an un-quantized signal r[k] and its quantized output b[k]. This is true of both sigma-delta DAC's and discrete-time sigma-delta ADC's. (A continuous-time sigma-delta is an exception since it does not operate on a simple output but instead filters its output and operates on that.)

How do we represent this in-band energy? With a low-pass filter, of course. Let's call the low-pass filter that describes what we mean by "in-band" as H(z). Ideally, H(z) will be a brick-wall filter:

$$H(e^{j\theta}) = \begin{Bmatrix} 1, \theta < B \\ 0, \theta > B \end{Bmatrix}, \; B = \frac{\pi}{OSR}$$

OSR is the over-sampling ratio. In actuality, most sigma-delta's are composed of accumulators and therefore have an n^{th} order response, typically 3^{rd}, 4^{th}, or 5^{th} order, depending on how much signal-to-

noise ratio is desired and how much over-sampling ratio is possible. (Let's be honest: the main driver of this factor is the project deadline and how confrontational your project management is.)

Hopefully, you have noticed the similarity here: The sigma-delta is trying to minimize

$$\int_{-\pi}^{\pi} \left| H(z)\big(B(z) - R(z)\big) \right|^2 ,$$

which according to Parseval, is equivalent to minimizing

$$\sum_{k=-\infty}^{k=\infty} |h[k] * (b[k] - r[k])|^2.$$

This is *almost* what the Viterbi algorithm does, minimizing

$$\sum_{k=-\infty}^{k=\infty} |h[k] * b[k] - r[k]|^2.$$

However, if r[k] is band-limited (i.e. does not have any components of out band B), then

$$h[k] * r[k] = r[k].$$

So, then we can write

$$\sum_{k=-\infty}^{k=\infty} |h[k] * (b[k] - r[k])|^2 = \sum_{k=-\infty}^{k=\infty} |h[k] * b[k] - r[k]|^2,$$

which the Viterbi algorithm *does* help us with.

Except for one problem: we need to place another constraint, that $h[k]$ needs to be a FIR filter. Recall that the Viterbi algorithm requires the error to be finite memory. As a result, we need h[k] to be a FIR filter, which means that it can't really be a brick-wall filter. However, with a FIR of long enough length, we can approximate a brick-wall to the point that it doesn't matter.

Caveats

Although it's fun (or at least mentally entertaining) to do the math, it is impractical to build this Viterbi detector in the discrete-time ADC case. The reason is that the input to the ADC needs to be discrete-time but un-quantized. It is the ADC's job to quantize that input. As a result, one must stay in the analog domain (or build a ridiculously high-precision flash sampler that would defeat the purpose).

In the case of the DAC, the approach makes sense. The information source is already digital. The sigma-delta DAC's job is to take a (relatively) unquantized signal and convert it to a more quantized signal that can then be run into a low-level flash sub-DAC.

Conclusion

In a communication system a transmitter takes a sequence of symbols $b[k]$, sends them through a channel modeled as a filter $h[k]$, resulting in a sequence r[k] received on the other end.

The receiver must then hypothesize out of all the possible b[k]'s that could have been sent to figure out which one *was* sent. To do so, it employs a Viterbi detector that minimizes the error between the hypothesized signal and the received signal.

The discrete-time sigma-delta ADC is almost identical to the receiver: it, too, receives a sequence of un-quantized samples and must then compute out of all the possible b[k]'s that minimize the (filtered) error, which one will minimize this error. There just doesn't happen to be a matched transmitter feeding it filtered values from a finite alphabet. Similarly, the sigma-delta DAC

Now, it doesn't matter in the sigma-delta case that no one was actually taking the b[k]'s and filtering them. In the end, both the communication receiver and the sigma-delta are minimizing the *same error function*.

About

Poojan Wagh is a contributing author at Circuit Sage. He has worked in the areas of direct-digital-RF PA/DAC's, CMOS GSM/EDGE transceivers, WCDMA continuous-time CMOS ADC's, CMOS TV tuners, and software-definable radio. He also runs his own circuit design blog at http://www.circuitdesign.info/blog.