Sigma-delta techniques extend DAC resolution

June 23, 2004 Embedded Staff

Motion control systems often require digital-to-analog converters with high resolution but not high accuracy. Because high-accuracy DACs are expensive and consume valuable board space, a good solution is to extend your DAC's resolution in software. This article presents such a technique.

Developers of closed-loop control systems often need a digital-to-analog converter (DAC) whose resolution far exceeds its accuracy. Indeed, when you've got good analog feedback sometimes you'll find that an 8-bit DAC has more than enough accuracy but not the 16 bits of resolution you need to attain your system performance goals.

The usual closed-loop control system uses a relatively inaccurate device yet achieves high accuracy through good feedback. While high DAC resolution is often necessary to keep the output steady, DAC inaccuracies are often insignificant compared to the device's own peculiarities.

Cost or size constraints might prevent you from adding a fancy high-precision, high-resolution DAC to your system. But you can steal a page from the audio industry and use a software sigma-delta modulator to extend the resolution of a cheap DAC or PWM output. When this technique works well it can give you enormous gains in effective DAC resolution for a cost-effective improvement in system performance.

Start dithering around

One easy way to increase a DAC's resolution is to dither its output. This is the whole idea behind a PWM drive. Figure 1 shows a DAC with a dithered output. The desired value is added to a sawtooth wave and the results are quantized and converted to analog by the DAC. With the correct arrangement of sawtooth generator and quantizer the DAC can give an average value that's closer to the desired output than what you can achieve without dithering.

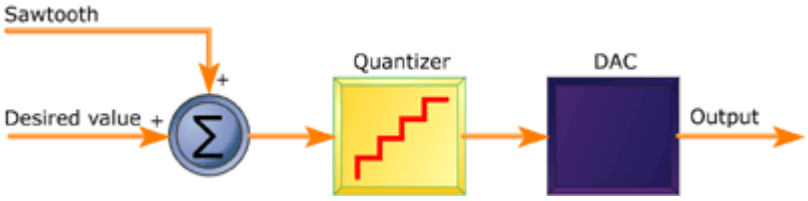

Figure 1: DAC with dithered output.

Figure 2 illustrates the behavior of a dithered DAC with a sawtooth period of $f/8$, where f is the basic sampling rate of the DAC process. In this figure the desired output is 1.625, and the DAC dithers between 1 and 2 to reach this average value. As you can see, the dithered DAC is just a pulse-width modulated device that varies its duty cycle to achieve the desired average value.

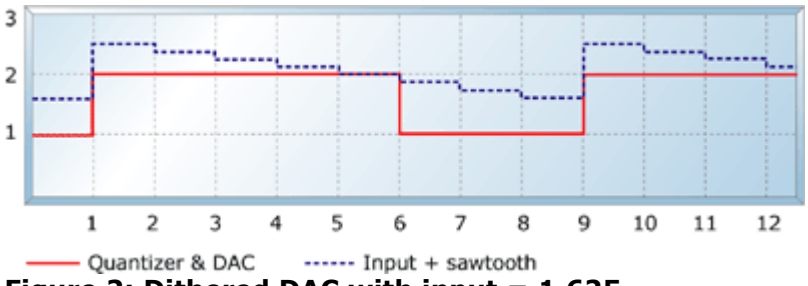

Figure 2: Dithered DAC with input = 1.625

The dithered DAC of Figure 1 has two main problems that can be overcome by a sigma-delta modulator. Because the fundamental division ratio of the sawtooth generator is fixed at design time the amount of added precision is also fixed. More important, the noise from the dithering tends to concentrate at the sawtooth frequency. Since this noise can be quite high it can bleed through whatever you're controlling and cause problems.

Both of these problems are reduced with a sigma-delta modulator, often to the point of insignificance. Figure 3 shows a basic first-order sigma-delta modulator. The desired output value is applied to the summing junction along with the actual output. This delta value is added to the integrator value (sigma) and applied to quantizer. The quantizer trims the data down to whatever the DAC is capable of representing.

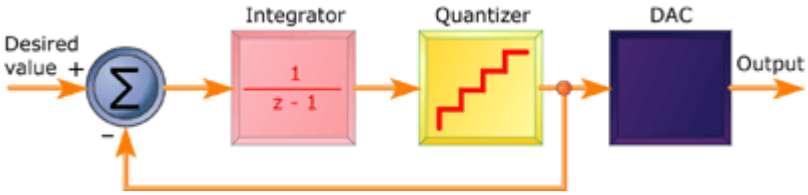

Figure 3: Basic sigma-delta modulator

With this arrangement the sigma-delta modulator automatically adjusts its output to ensure that the average error at the quantizer output is zero. This is due to the integrator in the signal path: the integrator value is the sum of all past values of the error, so whenever there is a non-zero error value the integrator value just keeps building until the error is once again forced to zero.

Like the "regular" dithered DAC shown in Figure 1, the sigma-delta modulator works because it dithers the quantized value sent to the DAC. Although the action of the sigma-delta modulator is inherent in its construction there's no forced behavior as with the dithered DAC. In the sigma-delta modulator the frequency of the dithered output adjusts itself automatically, and any "odd man out" pulses are automatically held to one cycle long.

Figure 4 shows how our sigma-delta modulator would work with the same 2-bit DAC of Figure 1 and an input of 1.625. Initially (Step 0), the integrator value is 1.625 and the quantizer passes 1 to the DAC, generating an error of 0.625. At Step 1 the integrator value goes to 2.25 and the DAC outputs a 2, generating an error of -0.325. The integrator and DAC cycle through the values shown, producing a DAC output with an average value of 1.625.

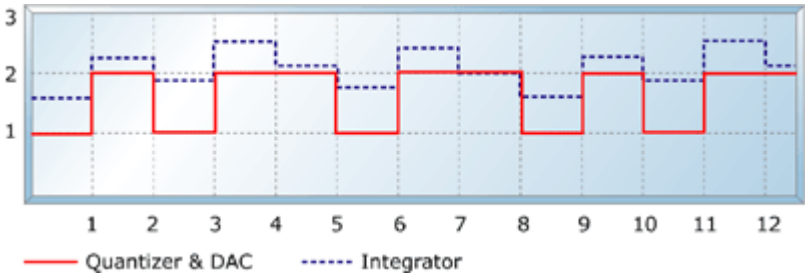

Figure 4: Sigma-delta modulator with input = 1.625

The difference between Figure 2 and Figure 4 points up the real advantage of the sigma-delta DAC. In Figure 2 the high and low portions of the cycle were all jammed together, which would give a controlled stage or a low-pass filter time to react to the (unwanted) dithering in the output. In Figure 4 the high and low portions of the cycle are spread out as much as possible. This means that whatever the DAC is driving will likely have a much lower response to the variations in the DAC output, so it will follow the average much better.

On averages

So far we've talked about having a DAC whose average error is kept to zero. In order to find out more about how and when to use a sigma-delta DAC we need to investigate this notion more formally.

Figure 5 shows the total error power for a 1-bit straight DAC compared to any dithered DAC (including a sigma-delta). Because the dithered DAC doesn't stay as close as possible to the desired value at all times the dithered DAC has significantly more error than the straight DAC for all input values except for 0, 0.5, and 1. Why, then, would we consider using any sort of dithered DAC at all?

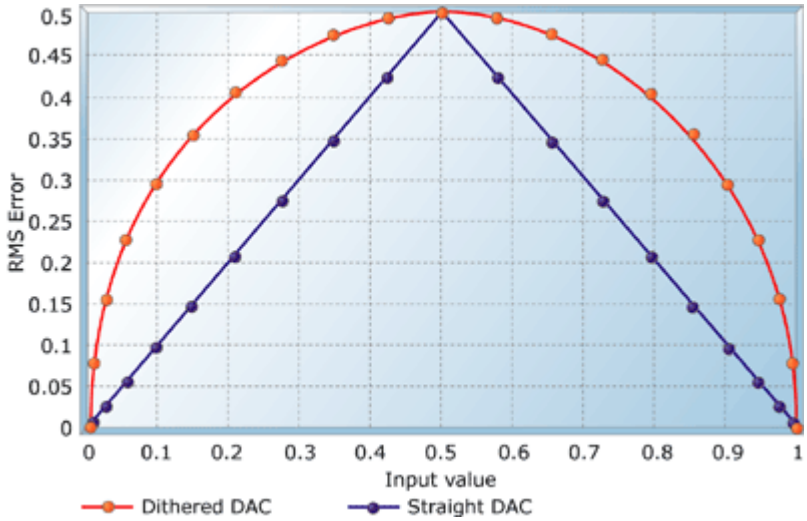

Figure 5: Total RMS Error for 1-bit "Straight" and dithered DACs

The answer is that we often care much more about the average DAC output than we care about its instantaneous value. In the case of the straight DAC the overall error is lower, but for a steady input the error is also steady and unavoidable. In frequency-domain terms, this means that we care more about resolution at low frequencies than we do about errors at high frequencies. This could be because we know that we can pass the output of the DAC through a low-pass filter, either because we know the hardware downstream of the DAC has an inherent low-pass characteristic or because the system is inherently insensitive to high-frequency errors.

Any dithered DAC will give a lower DC error than a straight DAC, but in the "regular" dithered DAC of Figure 1 the error energy will be at the dithering frequency and its harmonics. This may be useful, but to get good resolution out of a dithered DAC we must dither the DAC at a relatively low frequency. Doing so can push the error energy down into frequencies low enough to cause problems. The sigma-delta DAC does a much better job of "shaping" the error energy into the higher frequencies.

Definitions for "error" get a bit hazy with sigma-deltas, but Figure 6 shows the approximate RMS error vs. frequency for the sigma-delta modulator of Figure 3. The interesting thing to notice is that unlike the "regular" dithered DAC, the sigma-delta modulator causes the noise to be roughly proportional to frequency at up to half the DAC's clock rate.

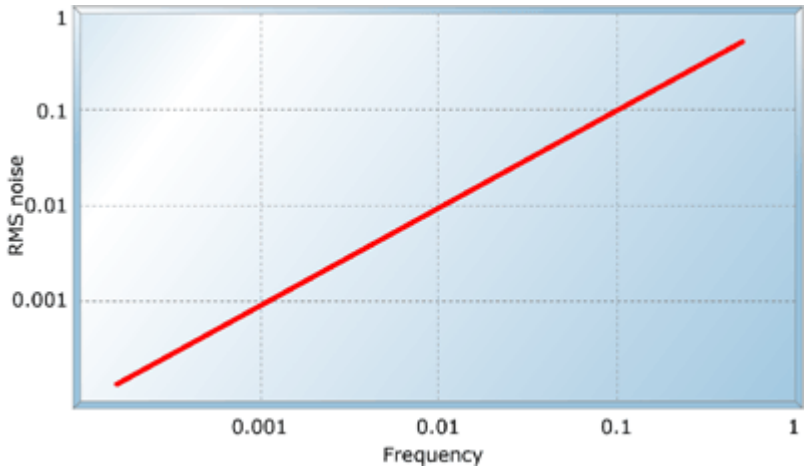

Figure 6: Sigma-delta noise spectrum

From Figure 6 you can see that any application that is insensitive to high frequencies will benefit from a sigma-delta DAC. Motors, heaters, and coolers, in particular, tend to have a strongly low-pass characteristic that makes them natural candidates for this technique. Any time you need low-frequency resolution and can update the DAC at much higher frequencies than those you're interested in, you can get benefit from a sigma-delta. Indeed, this is how an audio sigma-delta (or "one-bit") DAC works: it's a dedicated chip that samples the sigma-delta loop at several megahertz to shape the error frequencies into the high audio and ultrasonic ranges. Since the human ear can't hear noise at 15kHz nearly as well as it can at 1.5kHz the converter makes good-sounding audio with a DAC that costs much less than a highly accurate traditional DAC.

Put to the test

To see how a home-brewed sigma-delta DAC might play out for a real product design, let's take two examples. In both cases we'll update the DAC at 8kHz, and assume a requirement for 10 bits of resolution, yet our actual DAC needs will be significantly different. Our first system is a drive that must phase-lock a rotating wheel with some external source. Our second example will produce telephone-quality audio.

For the rotating-wheel example assume that we're sampling the control system at 100Hz. Assume for the sake of argument that with the DAC on full the motor accelerates at 2 10^6 degrees per seconds^2 , and that the motor must keep its wheel positioned to within ½ degree of the ideal position.

With a control system sample time of 100Hz at full DAC output the motor accelerates 200 degrees per sample^2 (where 1 sample = 1/100th of a second). This means that with an 8-bit DAC the motor will accelerate about 0.8 degrees per sample2 per DAC count. To stay well within our error budget we'd need a 10- or 12-bit DAC.

If we use a sigma-delta modulator running at 8kHz, however, we can extend our DAC resolution dramatically. With an 8kHz sample rate, the motor acceleration at the DAC full scale is 0.031 degrees per sample^2 . This implies that not only can the motor be satisfactorily controlled with an 8-bit DAC, but we could even get away with using a 1-bit DAC!

For the audio system assume that we want to have telephone-quality sound, with the noise power held to no more than 60dB below max power and a bandwidth of 3kHz. In this case a sigma-delta converter won't do us any good at all, because we need good data almost all the way out to the Nyquist frequency of the DAC. For this system we'd need to bite the bullet and use a 10-bit DAC or sample much faster if we want to use a sigma-delta converter (such as the industry-standard sigma-delta audio converters, which sample in the megahertz region).

Although sigma-delta converter theory can be subtle and mind-bending to understand, the implementation of a first-order sigma-delta modulator is dirt simple. The easiest way to make a sigma-delta is to increment a variable with the commanded number, then subtract the number that's actually applied to the DAC. This works whether you're using a 1-bit DAC, a 16-bit DAC or an odd-numbered counter in a processor's PWM output circuit.

Listing 1: A simple sigma-delta modulator

```
// A 1st-order sigma-delta modulator. This returns a value that
// is shifted down by 'shift' bits
int sigDelt(int in; int * state; unsigned shift)
{
  int retval;
  *state += in;
  retval = state >> shift;
  *state -= retval <>
  return retval;
}
```

Listing 1 shows a sigma-delta modulator. Values are input in extended precision, kept track of in the variable pointed to by **state** , and the return value is effectively shifted down by **shift** bits. For a given input value the return value will dither by 1 least significant bit with a duty cycle determined by the value of the first **shift** bits in the input.

Listing 2: A faster sigma-delta modulator

```
// A 1st-order sigma-delta modulator. This returns a value that
// is shifted down by the number of bits in a short int.  Note
// that this code will only work on a machine that fits two or
// more short ints into a long int, and that it assumes
// big-endian addressing.
typedef union USigDelt
{
  short int get[2];
  long  int put;
} USigDelt;
```

```
short int sigDelt(long int in; USigDelt * state)
{
  state->get[1] = 0;
  state->put += in;
  return get[1];
}
```

If you're willing to trade obfuscation and inflexibility for execution speed, you can use code like that in Listing 2. This code isn't portable, it won't work on all processors (for example any chip where “short” and “long” are the same size), and there are a number of variations that must be used depending on the processor. The general idea, however, is effective on 8-bit, 16-bit, and many 32-bit processors.

If you have an application that needs more resolution than you can get from small or affordable DACs, and you have the spare processor power to run it, a sigma-delta modulator can often save your day.

Tim Wescott owns Wescott Design Services, where he works on embedded motion control and general embedded systems design. You can reach him at .