techXclusives

Digitally Removing a DC Offset

(or "DSP Without Math?") - Part 1

By Ken Chapman
Staff Engineer, Core Applications - Xilinx UK

Introduction

Digital Signal Processing (DSP) - Does the very mention of this topic make you think of horrible mathematics and make you have the burning desire to dive for cover? If it does, then maybe you should approach the whole subject the way I started to about 8 years ago. Rather than start with mathematics, try to understand how simple functions in the analogue world can be modeled, and then convert them into digital representations.

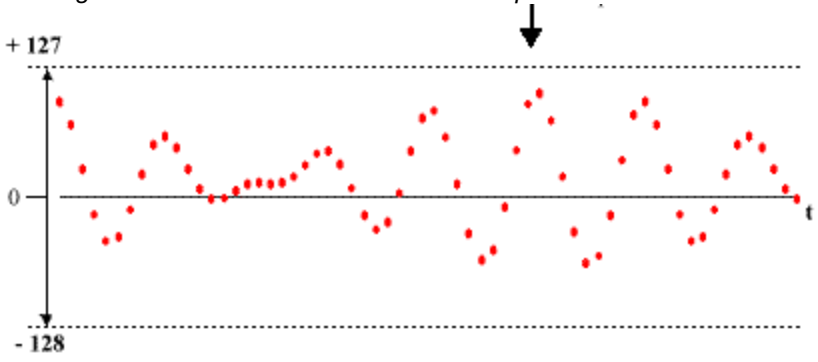
In Parts 1 and 2 of this *techXclusive*, I am going to examine how to remove the DC content from a digitally sampled waveform using DSP without mathematics -- well, nothing difficult, anyway! The first article will take a very gentle look at DSP and illustrate how to create a circuit capable of performing the required signal processing. In the second article, I will look at how to optimise the derived function for use in audio telecom applications, using my favorite SRL16E mode.

Sampled Waveforms

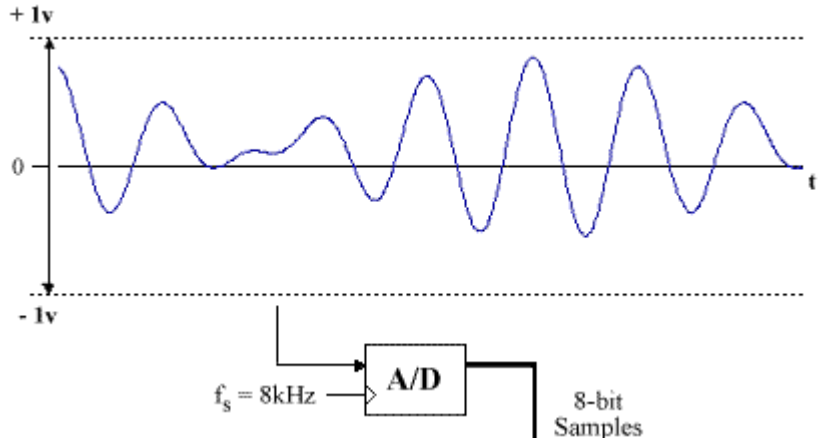
So, let's start by taking a look at the input signal. You may remember that DSP is all about digital samples, which are numbers that represent the amplitude of a waveform taken at regular intervals. These are normally the result of an Analogue-to-Digital Converter (A/D or ADC) that generates values of a given number of bits (resolution) at a sample rate set by a sample clock.

In the upper plot, we see an analogue waveform being applied to the A/D. The input signal needs to remain within the specified input range voltage swing of the A/D converter (in this case, ±1 volt). The A/D will sample this waveform at a frequency (fs) that is relatively fast in comparison to the frequency content of the signal. (Remember that Nyquist says you have to sample at a rate at least twice that of the highest frequency present?)

Test signal formed of 800Hz and 960Hz components with a DC offset

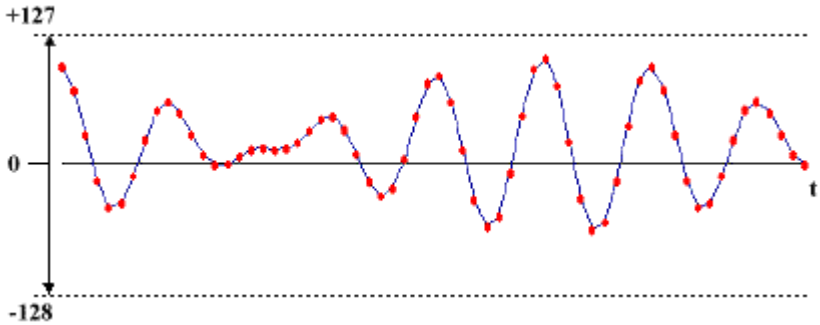


"



In the lower plot, the actual digital samples provided by the A/D are shown as red dots. The most important part of DSP has happened! In this case, the samples are represented by 8-bit numbers. A twos complement format is used in order to represent both positive and negative quantities. The first values of the plot are +104, +80, +31, -19, -48, and -44. We know that these can be represented in hexadecimal by 68, 50, 1F, ED, D0, and D4. So, all that DSP has to do is work with such a stream of numerical data and manipulate it in some way.

Although the only information you have to work with are the values represented by the red dots, the shape of the waveform can be made more apparent by joining the dots with straight lines. Now we can once again "see" the analogue waveform; of course, it is less pure than it originally was (the essence of quantisation noise).



Looking at the waveform plots, we can see that they contain high frequency components of some kind (800Hz and 960Hz in this test case). We also see that the waveform spends a greater percentage of time above the zero axis than it does below it, indicating some kind of positive DC bias. This is clearly seen in the digital plot, as there are many more positive red dots than negative.

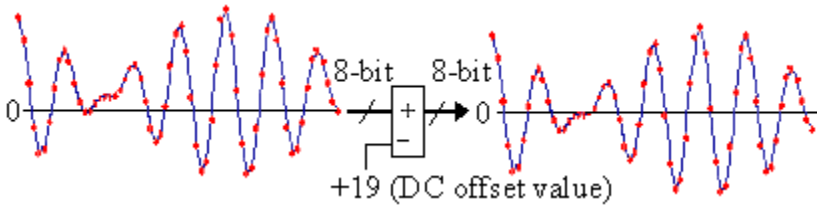
Generally speaking, a DC offset such as this is undesirable, because it tends to mean that the positive peaks of the waveforms are more likely to exceed the maximum level that can be represented. In the ideal world, the DC offset would be removed before the A/D conversion; however, this may be difficult to achieve. Indeed, it may be that the analogue components have unintentionally inserted the DC bias as part of the signal amplification and conditioning.

Removing DC Offset

Now, given that a DC offset has a frequency of zero, the DC offset can, theoretically, be removed by the use of a high pass filter. This may lead us directly into the world of serious DSP and investigation of such things as FIR (Finite Impulse Response) filters. However, I would like to take a more empirical approach to solving this problem – let's avoid as much math as

empirical approach to solving this problem -- let's avoid as much math as possible and hopefully find a much easier and cost-effective implementation!

If I know what the DC offset level is, then it is possible to remove it with a simple subtraction. As if by magic, I have now determined that the DC offset of the digital samples is equivalent to the value "+19". So, all we have to do is take each input sample from the A/D converter and subtract the value 19 from it. The output from the subtractor is the waveform without any DC offset. We have just performed DSP, because we have manipulated a stream of digital samples to form new samples. The value of the first sample is, of course, +104 -19 = +85.

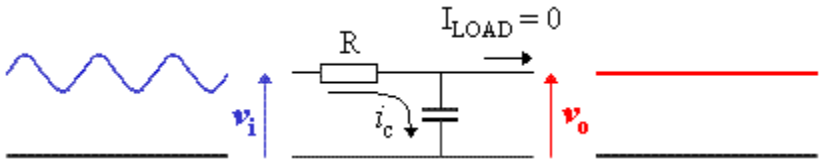


It may seem hard to believe that a subtractor can be an important DSP function, but it clearly is! (It is also very well-supported by Virtex™ and Spartan-II™ devices.) As with the basic adder function (see the previous *techXclusive* "8×12 Does NOT Equal 12×8"), each "slice" of the device can implement a 2-bit subtractor. Therefore, a simple 8-bit subtractor will require 4 "slices". As this is such a basic function, it is supported well in many design flows, including HDL and System Generator.

Finding the DC Level

Although we have achieved our objective to remove the DC offset with the subtractor, the process relied on me stating the DC value that was to be removed. Clearly, we now need to find a way to derive that DC offset value automatically. Although this will be a little more complicated, we can again take an empirical approach to finding a solution with another very common function, and avoid a lot of "DSP theory".

In the analogue world, the simplest way to find the average DC level of a signal is to smooth it with a capacitor. The larger the value of the smoothing capacitor, the steadier the DC level (especially if there is a load current).



In theory, we need to look at a differential equation to solve this simple circuit. However, if we take an instant in time, then simple linear equations can be used:

Voltage across the resister "R" is given by: $v_i - v_o$.

Therefore, the current flowing into the capacitor is given by:
$i_c = (v_i - v_o)/R$.

If the input voltage is higher than the average value "v_o", the capacitor will charge. Likewise, if the input voltage is lower than the average value, the current will be negative (flowing out of the capacitor), and the capacitor will discharge.

techXclusives

Digitally Removing a DC Offset
(or "DSP Without Math?") - Part 2

By Ken Chapman
Staff Engineer, Core Applications - Xilinx UK

Introduction

Welcome to the second part of this TechXclusive! Were you were able to spot the first optimisations that I am going to make to the circuit?

After the optimisations are made to the algorithm, I will focus on how to make a smaller and more efficient version for audio telecom applications. This is pure hardware engineering, and I don't feel that any design is complete until it has an SRL16E in it! I suggest you read my previous TechXclusive articles about the SRL16E before you read the last section of this one.

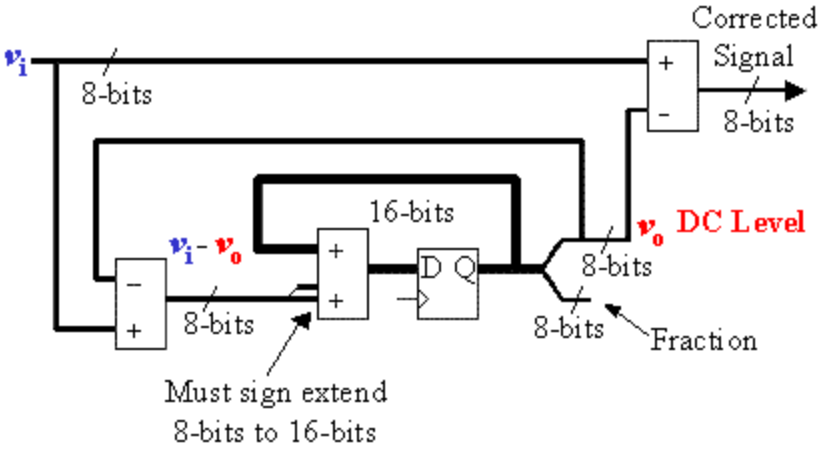
Removing the Multiplier Logic

Potentially, the largest part of the circuit so far is the multiplier. Although you may have some dedicated multipliers to spare in future Virtex-II designs, it is likely that the multiplier is costing slices in most cases. For really cost-sensitive designs with Spartan-II, we should be doing everything possible to reduce size and stay in the smallest device.

It is actually very easy to remove the multiplier from this circuit, and hopefully you have already seen how it can be achieved. I left a rather big clue on the two response plots in the last article by specifying the coefficient values as k=1/32 and k=1/256. In both cases, these values are represented by numbers in which only one bit is active. I am going to adopt the second of these values as I consider that the low ripple is much more desirable than the response time -- especially as even 100ms is relatively short.

```
    77                 01001101
  × 0.0039           ×      0.00000001
  --------           -----------------
    0.3003                  0.01001101
```

Since the multiplication process only requires that the variable input be multiplied by "1," the output product is the same as the input. Clearly, there is no need for a real multiplier; the output product is the same value, and the bit width is the same as the variable input. All that is required is to apply the variable input value with the binary point reassigned to the correct position. Our complete DC offset removal circuit is then reduced to the following logic...



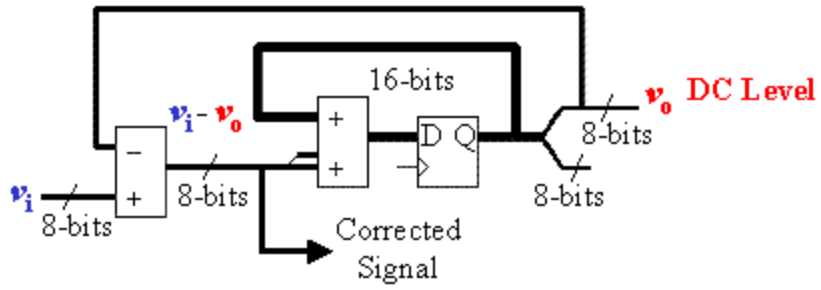
The circuit now consists only of an accumulator and two subtractors. Care is required when connecting the variable difference signal (vi - vo) to the accumulator input. To represent that the 8 bits are all fractional (to the right of the binary point), they are applied to the least significant byte of the 16-bit input. However, the upper byte must also be defined, and this must be achieved using sign extension (replicate the MSB of the 8-bit value a further 8 times to form either "00" or "FF" hexadecimal); this is so that the twos complement logic of the accumulator will correctly add both positive and negative values.

With this very small "k" value, it is obvious why the accumulation of the fractional parts (as well as the integer parts) must be performed.

Removing a subtractor

Having drawn out the DC level detector and the DC removing subtractor on the same diagram for the first time, it may appear more obvious that one of the subtractors is redundant. The corrected signal is the original signal with the DC level subtracted from it. This means that the output is the value vi - vo, which is the same as the difference signal being created by the subtractor within the DC detection circuit.

This means that the complete DC offset removal circuit can be reduced to just one accumulator and one subtractor...



Using the simple but accurate rule that a 2-bit add or subtract function fits into a "slice," then this circuit now only requires 12 "slices." For each additional bit of sample width, the subtractor and accumulator will each increase by 1-bit and, accordingly, increase the total size by 1 "slice." Therefore, with 16-bit input samples, the size would increase to 20 "slices." As a parallel circuit, this can also support a sample rate well in excess of 100MHz.

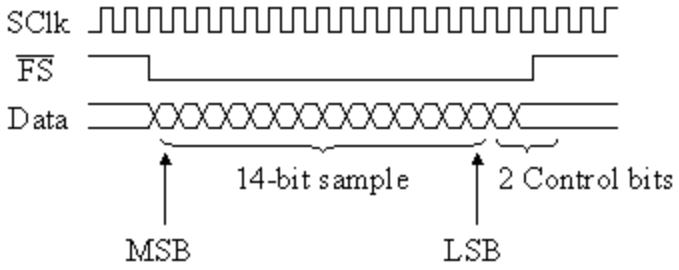
Low Sample Rate Applications

Although 12 and 20 "slices" may not sound like very much, it is still 6% to

10% of the smallest XC2S15 device, and this DC offset removal may well be seen as just a pre-process to the main function.

In a typical application, this DC offset removal may be required in a telephone conferencing facility. Each of the input lines (represented by digital samples) will ultimately be summed together within the system, and the contribution of multiple small DC offsets may have an adverse effect on the overall dynamics. The requirement for a DC offset removal circuit on each line input would soon see all those 6-10% units of a device mounting up to something significant.

Also typical of audio telecommunications, data samples are transmitted serially between units. These may be as packets within data frames, or even directly from the A/D converter. The Texas Instruments TLC320AC01C analogue Interface circuit (AIC) device uses a serial communications protocol with 14-bit A/D samples being transmitted with the most significant bit first as part of each 16-bit transfer...



To use the parallel implementation of the DC offset removal circuit, such serial data samples would need to be applied to a 14-bit shift register in order to read the sample in parallel. This requires an additional 7 "slices."

The linear equation for the charge on a capacitor is: Q = C.V = I.T

So, for a constant current of "I" for a period of time "T", the voltage on the capacitor will rise by the amount V = (I.T)/C

Obviously, the larger value of C, then the smaller the change in voltage for a given current and time.

This means that we can derive a final formula that describes this simple RC circuit:

$$\nabla V_o = [\nabla T / (R.C)]. (v_i - v_o)$$

OK, I said I wasn't going to do mathematics, but this really is a simple linear equation. During a period of time "∇T", the voltage across the capacitor will change by an amount proportional to the difference between the input voltage and the output voltage. This equation is only valid if the duration of time "∇T" is so small that the voltage change "∇V_o" does not significantly change the value of "$(v_i - v_o)$".

The equation can be simplified to $\nabla V_o = k. (v_i - v_o)$ by using a constant value "k", which is set by the combination of "R", "C", and the period of time over which each calculation is made "∇T". At the end of each period, the output voltage becomes the previous value of "V_o" plus the incremental value "∇V_o".

$$V_o' = V_o + k. (v_i - v_o) \qquad \text{where } k = [\nabla T / (R.C)]$$

We can try a simple experiment to prove this formula. Setting values of R=50Ω, C=100µF (a time constant of RC=5ms), and ∇T=1ms, then k=0.2. By applying a simple 10-volt step input, we can see that the calculated output for each step of time generates the exponential charge curve we would expect for such an RC circuit: