Efficient Resampling Implementations

by Douglas W. Barker

FPGA IMPLEMENTATION ISSUES

The structure of Figure 1(a) is easily implemented in hardware using a field programmable gate array (FPGA). The FPGA clock rate can be fixed at the $f_{s,in}$ clock rate in the case of the downsampler, or the $f_{s,out}$ clock rate in the case of the upsampler. No flip-flops in the design are required to operate on any other clock; this leads to a very efficient and elegant solution. Both resampling operations can take advantage of the hardware multipliers now available in FPGA's and RAM blocks can be used for coefficient storage. The number of hardware multipliers available impose a practical limitation on R, while the number of RAM blocks impose a practical limitation on L. For the case of the upsampler, ROM blocks may be used to store the coefficients even if the input sample rate is variable, whereas for the downsampler, RAM blocks must be used if one desires to change the output sample rate. In this case, an external microprocessor must calculate a new H every time the nominal output sample rate is to be changed.

Another peculiarity of the implementation is that for interpolation, the complexity of the prototype FIR filter does not change when the ratio of input sample rate to output sample rate changes. This is a consequence of the fixed interpolation factor. On the contrary, for decimation, larger sample rate ratios lead to higher filter complexity, that is, one must increase R proportional to the sample rate ratio in order to achieve the same filter specifications. A simple implementation trick can be used here to alleviate this problem to a certain degree. It is noted that when the sample rate ratio, $f_{s,out}/f_{s,in}$, is less than 0.5, we can reuse all R multipliers again for each computed output sample. This allows us to double R for ratios less than 0.5, while not increasing the number of hardware multipliers at all. It should be noted that we must double the number of coefficients in H since R has doubled, but we should also note that L' drops in half so the net affect is that there is no change in coefficient storage requirements. Similarly, when the ratio is less than 0.25, we may quadruple R as we can at every octave change in sample rate ratio. There is, however, one side effect, which is the length of the tapped-delay line in Figure 1(a); this will double, quadruple, etc., and hence poses practical limitation on the sample rate ratio.