Kishore A. Kotteri, Amy E. Bell, and Joan E. Carletta

Quantized FIR Filter Design Using Compensating Zeros

This article presents a design method for translating a finite impulse response (FIR) floating-point filter design into an FIR fixed-point multiplierless filter design. This method is simple, fast, and provides filters with high performance. Conventional wisdom dictates that finite word-length (i.e., quantization) effects can be minimized by dividing a filter into smaller, cascaded sections. The design method presented here takes this idea a step further by showing how to quantize the cascaded sections so that the finite word-length effects in one section are guaranteed to compensate for the finite word-length effects in the other section. This simple method, called "compensating zeros," ensures that: 1) the quantized filter's frequency response closely matches the unquantized filter's frequency response (in both magnitude and phase); and 2) the required hardware remains small and fast.

Digital filter design typically begins with a technique to find double-precision, floating-point filter coefficients that meet some given performance specifications—like the magnitude response and phase response of the filter. Two well-known techniques for designing floating-point FIR filters are the windowing method and the equiripple Parks-McClellan method [1], [2]. If the filter design is for a real-time application, then the filter must be translated to fixed point, a more restrictive form of mathematics that can be performed much more quickly in hardware. For embedded systems applications, a multiplierless implementation of a filter is advantageous; it replaces multiplications with faster, cheaper shifts and additions. Translation to a fixed-point, multiplierless implementation involves quantizing the original filter coefficients (i.e., approximating them using fixed-point mathematics). The primary difficulty with real-time implementations is that this translation alters the original design; consequently, the desired filter's frequency response characteristics are often not preserved.

Multiplierless filter design can be posed as an optimization problem to minimize the degradation in performance; simulated annealing, genetic algorithms, and integer programming are among the many optimization techniques that have been employed [3]. In general, however, optimization techniques are complex, can require long run times, and provide no performance guarantees. The compensating zeros technique is a straightforward, intuitive method that renders optimization unnecessary; instead, the technique involves the solution of a linear system of equations. It is developed and illustrated with two examples involving real-coefficient FIR filters; the examples depict results for the frequency response as well as hardware speed and size.

Quantized Filter Design Figures of Merit

Several important figures of merit are used to evaluate the performance of a filter implemented in hardware. The quantized filter design evaluation process begins with the following two metrics.

▲ 1) Magnitude mean-squared-error (MSE) represents the average of the squared difference between the magnitude response of the quantized (fixed-point) filter and the unquantized (ideal, floating-point) filter over all frequencies. A linear phase response can easily be maintained after quantization by preserving symmetry in the quantized filter coefficients.

▲ 2) Hardware complexity. In a multiplierless filter, all mathematical operations are represented by shifts and additions. This requires that each quantized filter coefficient be expressed as sums and differences of powers of two: for each coefficient, a representation called canonical signed digit (CSD) is used [3]. CSD

"DSP Tips & Tricks" introduces practical tips and tricks pf design and implementation of signal processing algorithms so that you may be able to incorporate them into your designs. We welcome readers who enjoy reading this column to submit their contributions. Contact Associate Editors Rick Lyons (r.lyons@ieee.org) or Amy Bell (abell.vt.edu).

1053-588/03/$17.00©2003IEEE

format expresses a number as sums and differences of powers of two using a minimum number of terms. Before a quantized filter design is implemented in actual hardware, the hardware complexity is estimated in terms of T, the total number of nonzero terms used when writing all filter coefficients in CSD format. In general, the smaller T is, the smaller and faster will be the hardware implementation. For application-specific integrated circuit and field programmable gate array (FPGA) filter implementations, a fully parallel hardware implementation requires $T-1$ adders; an embedded processor implementation requires $T-1$ addition operations.

Once the filter has been implemented in hardware, it can be evaluated more directly. Important metrics from a hardware perspective include: hardware size, throughput (filter outputs per second), and, latency (time from filter input to corresponding filter output). The relative importance of these metrics depends on the application.

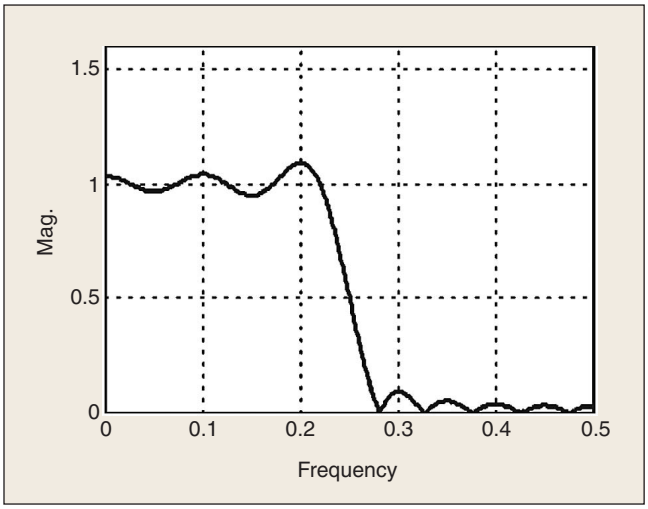
The goal of the quantized filter design is to achieve a small magnitude MSE while keeping the hardware costs low. In general, the higher the value of T, the closer the quantized filter coefficients are to the unquantized coefficients and the smaller the magnitude MSE. Conversely, smaller T implies worse magnitude MSE. Hence, there is a tradeoff between performance and hardware cost; T can be thought of as the parameter that controls this tradeoff.

Filter Structures

Filter designs can be implemented in hardware using various structures. The three most common structures are: direct, cascade, and lattice. In general, pole-zero, infinite impulse response (IIR) filters are more robust to quantization effects when the cascade and lattice structures are employed; performance degrades quickly when the direct structure is used [1], [2].

For all-zero, FIR filters, the direct structure usually performs well (if the zeros are not very clustered but are moderately uniformly distributed) [1], [2]. Moreover, since most FIR filters have linear phase (the filter coefficients are symmetric), the lattice structure cannot be used because at least one reflection coefficient equals ± 1. Although the direct structure is a good choice for many FIR filter implementations, the cascade structure offers at least one advantage. When an FIR filter is quantized using a direct structure, the quantization of one coefficient affects all of the filter's zeros. In contrast, if an FIR filter is quantized with a cascade structure, the quantization of coefficients in one of the cascaded sections affects only those zeros in its section—the zeros in the other cascaded sections are isolated and unaffected. Depending on the application, it may be important to more closely approximate the unquantized locations of some zeros than others.

The compensating zeros method uses a cascade structure. However, it goes beyond a "simple quantization" technique that uniformly divvies up the given T nonzero terms in CSD format across the coefficients in the cascaded sections. The next section first illustrates a simple quantization approach for an FIR filter design using a cascade structure; then the compensating zeros method is developed and used to redesign the same FIR filter. The result is an improvement in the magnitude MSE for the same T.



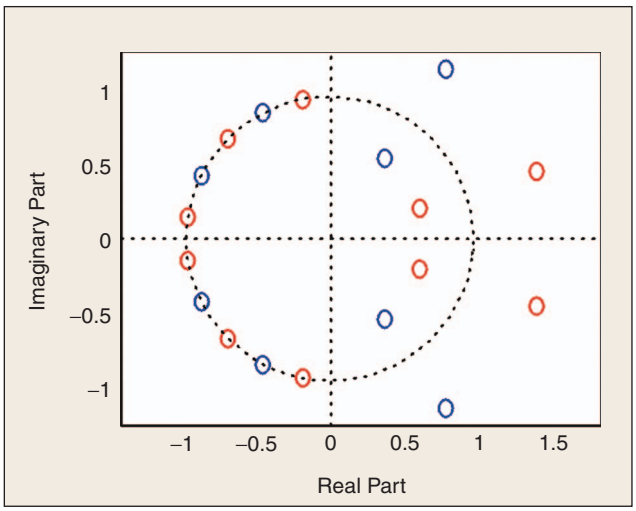
▲ *1. Frequency magnitude response of h(n) for the windowed FIR filter.*

Example 1: A Windowed FIR Filter

Consider a low-pass, symmetric, length-19 FIR filter designed using a rectangular window. The filter has a normalized (such that a frequency of one corresponds to the sampling rate) passband edge frequency of 0.25 and exhibits linear phase. The floating-point filter coefficients are

Table 1. Unquantized windowed FIR filter coefficients, *h*(*n*).

n	*h*(*n*)
0, 18	0.03536776513153
1, 17	–1.94908591626e-017
2, 16	–0.04547284088340
3, 15	1.94908591626e-017
4, 14	0.06366197723676
5, 13	–1.9490859163e-017
6, 12	–0.10610329539460
7, 11	1.94908591626e-017
8, 10	0.31830988618379
9	0.50000000000000



▲ *2. Pole-zero plot for h(n). The zeros are divided into two cascaded sections by placing the blue zeros in the first section $c_1(n)$, and the red zeros in the second section $c_2(n)$.*

listed in Table 1, and Figure 1 shows the unquantized magnitude response of this filter.

Figure 2 illustrates the pole-zero plot for $h(n)$. To implement this filter in the cascade form, $h(n)$ is split into two cascaded sections whose coefficients are $c_1(n)$ and $c_2(n)$. This is accomplished by distributing the zeros of $h(n)$ between $c_1(n)$ and $c_2(n)$.

To separate the zeros of $h(n)$ into the two cascaded sections, the z-plane is scanned from $\omega = 0$ to $\omega = \pi$. As they are encountered, the zeros are placed alternately in the two sections. The first zero encountered is at $z = 0.66e^{0.324j}$. This zero, its conjugate, and the two reciprocals are put in one section. The next zero at $z = 0.69e^{0.978j}$, its conjugate, and the reciprocal pair are placed in the other section. This proceeds until all of the zeros of the unquantized filter are divided among the two cascade sections.

The steps in this zero-allocation process are as follows: compute the roots of $h(n)$; partition those roots into two sets of roots as described above; and determine the two sets of coefficients, $c_1(n)$ and $c_2(n)$, for the two polynomials associated with the two sets of roots. The section with fewer zeros becomes the first section in the cascade, $c_1(n)$, and the section with more zeros becomes the second section, $c_2(n)$ (this approach provides more degrees of freedom in our design method—see Design Rule-of-Thumb number 6).

For the example, the zero allocation is illustrated in Figure 2 where the eight blue zeros go to $c_1(n)$, which has length nine, and the ten red zeros go to $c_2(n)$, which has length 11. This method of splitting up the zeros has the advantage of keeping the zeros relatively spread out within each section, thereby minimizing the quantization effects within each section. Because complex conjugate pairs and reciprocal pairs of zeros are kept together in the same cascade section, the two resulting sections have symmetric, real-valued coefficients. The resulting floating-point cascade $c_1(n)$ and $c_2(n)$ coefficients are shown in Table 2.

Figure 3 depicts the block diagram corresponding to $h(n)$ and the equivalent cascaded form. Coefficients $c_1(n)$ and $c_2(n)$ are normalized so that the first and last coefficients in each section are one; this ensures that at least two of the coefficients in each cascade section are efficiently represented in CSD format. Consequently, it is necessary to include a gain factor, k in Table 2, following the cascade sections. The magnitude response of the cascaded filter is identical to Figure 1.

Now consider a fixed-point, quantized, multiplierless design of this cascade structure so that it can be implemented in fast hardware. Assume that there are a fixed total number of CSD terms, T, for repre-

Table 2. Unquantized and simple-quantized cascaded coefficients for $h(n)$, ($T = 25$).

$c_1(n)$		$c_2(n)$		$c_1'(n)(T=9)$		$c_2'(n)(T=14)$	
n_1		n_2		Decimal	CSD	Decimal	CSD
0, 8	1.0000000	0, 10	1.0000000	1.00	001.00	1.000	001.000
1, 7	0.3373269	1, 9	−0.3373269	0.25	000.01	−0.250	000.0$\underline{1}$0
2, 6	0.9886239	2, 8	−2.1605488	1.00	001.00	−2.000	0$\underline{1}$0.000
3, 5	1.9572410	3, 7	−0.8949404	2.00	010.00	−1.000	00$\underline{1}$.000
4	3.0152448	4, 6	1.8828427	4.00	100.00	1.875	010.00$\underline{1}$
		5	3.5382228			3.500	100.$\underline{1}$00
	$k = 0.0353678$			$k' = 0.03515625$		0.00001001 ($T = 2$)	

senting the two unquantized cascaded sections and the unquantized gain factor. Two different techniques are considered for quantizing the filter: a simple quantization method that treats each filter section independently, and our proposed compensating zeros method in which the quantization errors in one section are compensated for in the next section.

Simple Quantization

For the simple quantization method, in the process of distributing a fixed number of CSD terms T to a single cascade section with n coefficients, all reasonable distributions are examined. These "reasonable distributions" consider all of the "mostly uniform" T allocation schemes to n coefficients: all coefficients receive at least one CSD term and the remaining CSD terms are allocated to those coefficients that are most different (in terms of percent different) from their unquantized values. Extremely nonuniform allocation schemes (e.g., one coefficient receives all of the T and the remaining coefficients are set to zero) are not considered.

Of all the distribution schemes examined, the distribution that gives the best result (i.e., the smallest magnitude MSE) is chosen. (Note: this does not require an optimization technique; for reasonably small values of T, it is simple to organize a search that looks in the area around the floating-point coefficients, which is the only area where high quality solutions lay.) This process ensures that there is no better simple quantization scheme for the given cascaded filter.

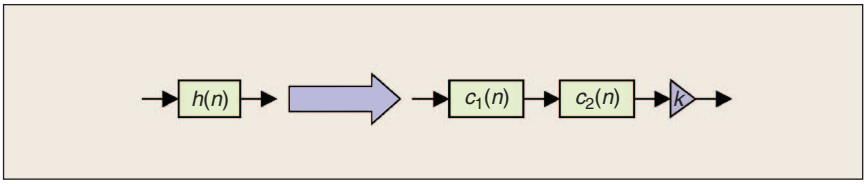
In applying the simple quantization method to the windowed FIR filter example, the unquantized cascade coefficients, $c_1(n)$ and $c_2(n)$, are independently quantized to the simple quantized cascade coefficients, $c_1'(n)$ and $c_2'(n)$. In this example, a total of $T = 25$ CSD terms was chosen; this choice results in small hardware while still providing a reasonable approximation to the desired filter. Based on the relative lengths of the sections, nine CSD terms are used for $c_1'(n)$, 14 terms are used for $c_2'(n)$, and two terms are used for the quantized gain factor k'. The resulting simple quantized coefficients are listed in Table 2 (in the CSD format, an underline indicates that the power of two is to be subtracted instead of added).

The frequency response of the simple quantized filter, $c_1'(n)$ and $c_2'(n)$, is compared to the unquantized filter, $h(n)$, in Figure 4. Although a linear phase response is retained after simple quantization (i.e., the simple quantized coefficients are symmetric), the magnitude response is significantly different from the original, unquantized case.

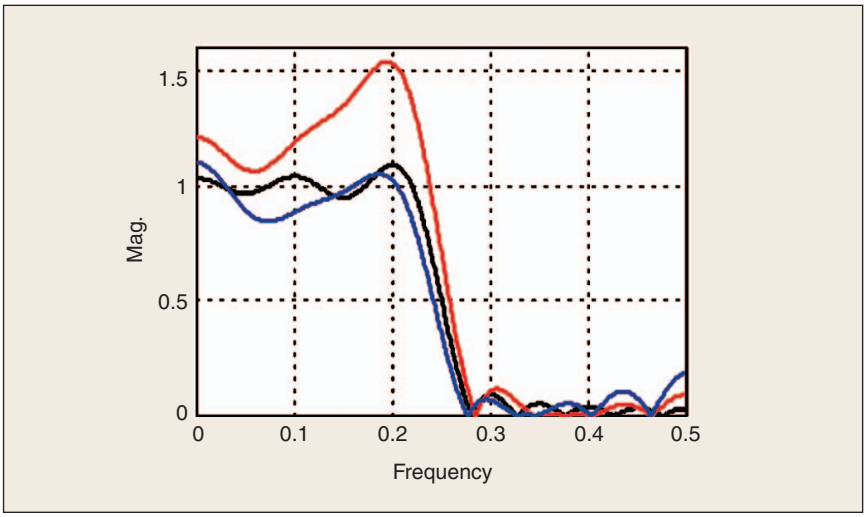
Compensating Zeros Quantization

The proposed compensating zeros quantization method takes the quantization error of the first cascaded section into account when quantizing the second section. The key to this method is the desire that the frequency response of the quantized cascade structure match the frequency response of the original, unquantized direct structure.

Quantization using the compensating zeros method begins with the quantization of the first cascade section $c_1(n)$ to $c_1'(n)$ and the gain factor k to k' (using the simple quantization method described in the previous section). Next, instead of quantizing $c_2(n)$ to $c_2'(n)$, $c_{\text{comp}}(n)$ is computed such that $c_1'(n)$ cascaded with $c_{\text{comp}}(n)$ is as close as possible to the original filter $h(n)$. Coefficients $c_{\text{comp}}(n)$ are called the compensating section, since their aim is to compensate for the performance degradation resulting from the quantization of $c_1'(n)$; the computation of $c_{\text{comp}}(n)$ is developed below.



▲ *3. Direct form of h(n) and the equivalent cascade form using $c_1(n)$, $c_2(n)$, and k.*



▲ *4. Frequency responses of the unquantized windowed filter h(n) (black), simple quantization (red), and compensating zeros quantization (blue).*

If $C_1(z)$, $C_2(z)$, $C_1'(z)$, $C_2'(z)$ and $C_{\text{comp}}(z)$ are the transfer functions of $c_1(n)$, $c_2(n)$, $c_1'(n)$, $c_2'(n)$ and $c_{\text{comp}}(n)$ respectively, then the transfer function of the unquantized cascaded filter $H(z)$ can be written as

$$H(z) = kC_1(z)C_2(z), \quad (1)$$

where k is the gain factor. The transfer function of the semiquantized filter using the compensating zeros method is given by

$$H'_{\text{comp}}(z) = k'C_1'(z)C_{\text{comp}}(z). \quad (2)$$

$H'_{\text{comp}}(z)$ is called the semiquantized filter because $c_{\text{comp}}(n)$ has floating-point coefficients. The goal is for the semiquantized and unquantized transfer functions to be equal, that is, $H'_{\text{comp}}(z) = H(z)$, or

$$k'C_1'(z)C_{\text{comp}}(z) = kC_1(z)C_2(z). \quad (3)$$

In (3), $C_1(z)C_2(z)$ and k on the right-hand side are the known, unquantized cascade filters. After $c_1(n)$ and k are quantized, $C_1'(z)$ and k' on the left-hand side are known. Thus, (3) can be solved for $c_{\text{comp}}(n)$.

Since the 11-tap $c_{\text{comp}}(n)$ is symmetric (with the first and last coefficients normalized to one), it can be expressed in terms of only five unknowns. In general, for a length-N symmetric filter with normalized leading and ending coefficients, there are M unique coefficients where $M = \lceil (N-1)/2 \rceil$. (The $\lceil x \rceil$ notation means: the next integer larger than x; or if x is an integer, $\lceil x \rceil = x$.)

Equation (3) can be evaluated at $M = 5$ values of z to solve for the five unknowns in $c_{\text{comp}}(n)$. Since the frequency response is the primary concern, these values of z are on the unit circle. For the example, (3) is solved at the frequencies $f = 0$, 0.125, 0.2, 0.3 and 0.45 (i.e., $z = 1$, $e^{j0.25\pi}$, $e^{j0.4\pi}$, $e^{j0.6\pi}$, $e^{j0.9\pi}$). MATLAB and MathCAD code used to solve the linear system of equations given by (3) to compute $c_{\text{comp}}(n)$ for the windowed FIR filter example is available at http://www.cspl.umd.edu/spm/tips-n-tricks/.

Table 3 lists the computed floating-point coefficients of $c_{\text{comp}}(n)$. Now that $c_{\text{comp}}(n)$ has been obtained, it is quantized (using simple quantization and the remaining T) to arrive at $c_2'(n)$. The final, quantized $c_2'(n)$ filter coefficients using this compensating zeros method are also given in Table 3. Thus the compensating zeros quantized filter coefficients are the $c_1'(n)$ and k' from Table 2 in cascade with the $c_2'(n)$ in Table 3.

The frequency response of the compensating zeros quantized filter is compared to the unquantized filter and the simple quantized filter in Figure 4; the overall frequency response of the compensating zeros quantized implementation is closer to the unquantized filter than the simple quantized implementation. The small value of $T = 25$ employed in this example hampers the ability of even the compensating zeros quantized magnitude response to closely approximate the unquantized magnitude response. The magnitude MSE for compensating zeros quantization (7.347e-3) is an order of magnitude better than the magnitude MSE for simple quantization (4.459e-2).

For a fixed value of T, there are two ways to quantize the cascaded coefficients: simple and compensating zeros. If T is large enough, then simple quantization can achieve the desired frequency response. When T is restricted, however, compensating zeros quantization provides an alternative that outperforms simple quantization. In the example, it turns out that for $T = 26$, the simple quantization method can achieve the same magnitude MSE as the $T = 25$ compensating zeros quantization method. This improvement is achieved when the

Table 3. Unquantized $c_{\text{comp}}(n)$ and compensating zeros-quantized $c_2'(n)$ for $h(n)$($T = 25$).

n	$c_{\text{comp}}(n)$	$c_2'(n)(T = 14)$	
		Decimal	CSD
0, 10	1.0000000	1.00	001.00
1, 9	–0.7266865	–0.75	00$\underline{1}$.01
2, 8	–1.1627044	–1.00	00$\underline{1}$.00
3, 7	–0.6238289	–0.50	000.$\underline{1}$0
4, 6	1.2408329	1.00	001.00
5	2.8920707	3.00	10$\underline{1}$.00

Table 4. Distance of the zeros in Figure 5 from the origin.

	$\lvert z_2 \rvert$	$\lvert z_1 \rvert$
Unquantized	0.66061185	0.69197298
Comp. Quantized	0.67774476	0.644315248

one extra T is assigned to the first cascaded section; no improvement is realized when it is assigned to the second cascaded section. There is an "art" in how to assign the extra T: these terms should be allocated to the coefficients—in either cascaded section—that are most different (in terms of percent different) from their unquantized values.

The compensating zeros quantization procedure is outlined as follows.

▲ Step 1. Derive the unquantized cascade coefficients, $c_1(n)$, $c_2(n)$ and k, from the given, direct unquantized coefficients, $h(n)$.

▲ Step 2. Quantize $c_1(n)$ to $c_1'(n)$ and k to k' using the simple quantization method.

▲ Step 3. Select a set of M unique positive frequencies. For symmetric filters, M is given by $M = \lceil (N-2)/2 \rceil$; for nonsymmetric filters, M is given by $M = \lceil (N-1)/2 \rceil$, where N is the length of the second cascaded section's $c_2(n)$.

▲ Step 4. Solve for the M unknown $c_{\text{comp}}(n)$ coefficients by solving (3). For symmetric filters, (3) is evaluated at the M positive frequencies selected in Step 3. For nonsymmetric filters, (3) is solved at M positive frequencies and the corresponding M negative frequencies.

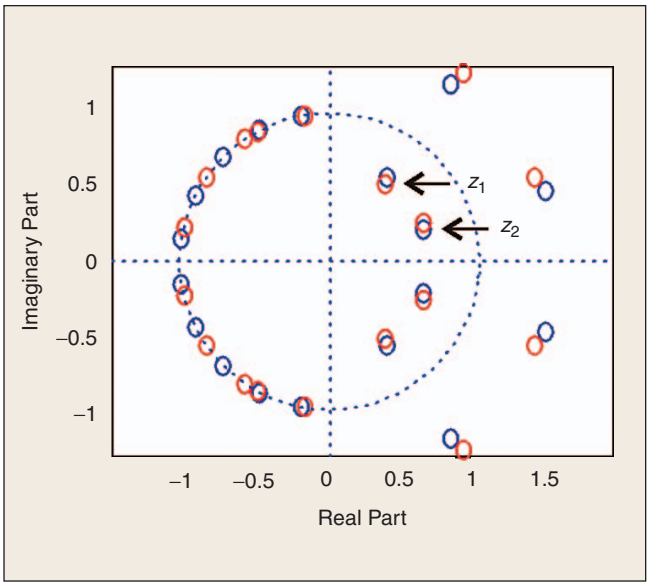
▲ Step 5. Using the simple quantization method, quantize the floating-point $c_{\text{comp}}(n)$ coefficients—with the remaining T from Step 2—to arrive at $c_2'(n)$.

▲ Step 6. Compute the direct form coefficients of the compensating zeros quantized filter using $H'(z) = k'C_1'(z)C_2'(z)$ (if desired).

Although the quantized $c_2'(n)$ values obtained using the compensating zeros method (in Table 3) are not very different from the values obtained using the simple method (in Table 2), it is important to note that simple quantization could not possibly find these improved coefficients. In the simple quantization case, the goal is to closely approximate the floating-point $c_1(n)$ and $c_2(n)$ coefficients in Table 2. However, in the compensating zeros quantization case, the goal is to closely approximate a different set of floating-point coefficients—$c_1(n)$ and $c_{\text{comp}}(n)$. Figure 5 compares the zero location plots of the compensating zeros quantized filter (red circles) and the unquantized filter (blue circles).

Figure 5 also shows the zeros z_1 and z_2 in the first quadrant; recall that z_1 is in $c_1(n)$ and z_2 is in $c_2(n)$. The quantization of $c_1(n)$ to $c_1'(n)$ moves z_1 away from the unit circle. Consequently, the compensating zeros quantization method moves z_2 toward the unit circle (Table 4 shows the distances of z_1 and z_2 from the origin before and after quantization). This compensating movement of the quantized zeros is the reason the quantized magnitude response more closely resembles the unquantized magnitude response; it also motivates the name of the quantization method.

The hardware performance of the simple quantized and compensating zeros quantized filters was evaluated by implementing the filters on an Altera FPGA. For each filter, given the desired CSD coefficients, filter synthesis software, written in C, was used to generate a synthesizable description of the filter in VHDL, a hardware description language. The software automatically chooses appropriate bit widths for internal signals such that errors due to truncation and overflow are completely avoided. In this way, an implemented fixed-point filter provides exactly the same outputs as the same filter would in infinite precision, given the quantized, finite precision filter



▲ *5. Zero plot comparing the unquantized zeros of h(n) (blue circles) and the compensating zeros quantized zeros of h′(n) (red circles).*

Table 5. Hardware metrics for the windowed FIR example.

	Simple quantized	Compensating zeros quantized
Hardware complexity (logic elements)	1,042	996
Throughput (Mresults/second)	82.41	83.93
Latency (clock cycles)	15	15
Input data format	(8, 0)	(8, 0)
Output data format	(23, −13)	(21, −12)

coefficients. Fixed-point, two's complement adders are used, but the number of integer and fraction bits for each hardware adder are chosen so that no round-off or overflow can occur.

The VHDL description generated is a structural one. A high performance, multiplierless implementation is achieved. For each section in a filter, a chain of registers is used to shift the data in, and the data is shifted in accordance with the filter coefficients before being summed. For example, if one of the filter coefficients were $18 = 2^4 + 2^1$, the corresponding data word would go to two shifters and be shifted four and one places, respectively, before being summed. A pipelined tree of carry save adders (CSAs) is used for the summation. The CSA tree produces two outputs that must be summed, or "vector-merged," to produce the final filter output. For the results presented here, we use a ripple carry adder for the vector merge, taking care to exploit special purpose routing ("carry chains") provided on Altera FPGAs to make ripple carry addition fast.

Table 5 summarizes the hardware performance of the filters. Data formats for all signals are shown as (n,l), where n is the total number of bits, including the sign bit, and 2^l is the weight of the least significant bit. Both filters take in inputs of data format (8,0), i.e., 8-bit two's complement integers. They vary in terms of their output data formats, depending on the precision of the coefficients used. Throughput is the most important performance metric; it measures how many inputs can be processed per second. Latency also bears on performance, but is less critical; it measures how many clock cycles a particular set of data takes to pass through the system, from input to corresponding output.

The results of Table 5 show that the compensating zeros quantized filter is slightly smaller and faster than the simple quantized filter. This is because the coefficients for the compensating zeros quantized case turn out to be slightly less wide (in terms of bits) than those for the simple quantized case, so that the adders also turn out to be slightly less wide.

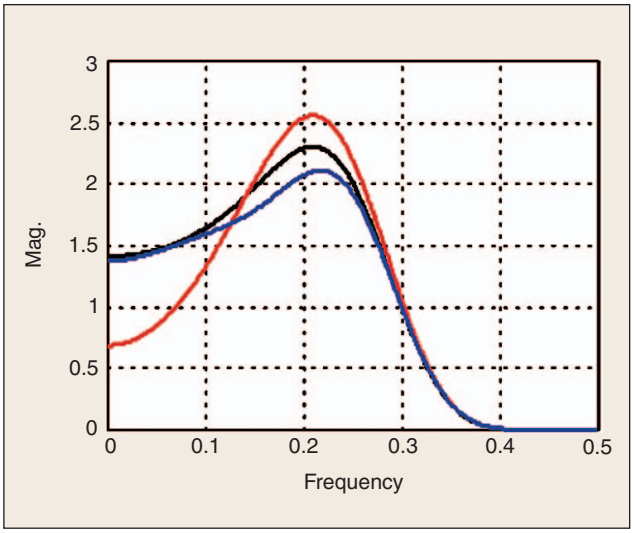
Example 2: A Biorthogonal FIR Filter

Now consider another example that illustrates the cascade structure's advantage of isolating some of the zeros in an FIR filter. The biorthogonal FIR wavelet filters are best known for their inclusion in the most recent version of the international image compression standard, JPEG2000 [4]; in this example, the 20-tap, biorthogonal, FIR, low-pass analysis wavelet filter is examined [5]. This filter has a passband edge frequency of 0.31 and exhibits linear phase. This filter has nine zeros clustered at $z = -1$ and ten zeros on the right-hand side of the z-plane.

For biorthogonal filters, the stopband magnitude response is characterized by the cluster of zeros at $z = -1$; it is advantageous to employ the cascade structure to place all of the zeros at $z = -1$ into an isolated cascade section. This cascade section will keep the zeros exactly at $z = -1$ have only integer coefficients and require no quantization. Furthermore, this one large cascade section can be split up into smaller T-friendly sections having four, two or one zeros to reduce the total T required. The remaining zeros on the right-hand side of the z-plane determine the passband characteristics and are separated into two cascade sections, $c_1(n)$ and $c_2(n)$, as before. Here the nine zeros at $z = -1$ are divided into two sections of four zeros, $c_3(n)$ and $c_4(n)$, and one section of one zero, $c_5(n)$.

Using the simple quantization method, the unquantized cascade sections $c_1(n)$ and $c_2(n)$ are quantized, but $c_3(n)$, $c_4(n)$, and $c_5(n)$ are captured exactly; a total of $T = 38$ CSD is chosen. The compensating zeros quantization method follows the procedure outlined above for $c_1(n)$ and $c_2(n)$; the remaining three cascaded sections are again captured exactly; a total of $T = 38$ CSD was used.

Figure 6 illustrates that the magnitude response of the compensating zeros quantized implementation is closer to the unquantized filter than the simple quantized implementation. The magnitude MSE for compensating zeros quantization (9.231e-3) is an order of magnitude better than the magnitude MSE for simple quantization (8.449e-2). For the biorthogonal FIR filter, it turns out that four extra T, ($T = 42$), are



▲ *6. Frequency response of the unquantized biorthogonal filter (black) compared to the two quantized filters: simple quantization (red) and compensating zeros quantization (blue).*

required for the simple quantization method to achieve the same magnitude MSE as the $T = 38$ compensating zeros quantization method.

Design Rules-of-Thumb

The following guidelines recommend how to derive the maximum benefit from the compensating zeros quantization technique.

▲ 1) As T increases for a given filter design, the performance of simple quantization approaches compensating zeros quantization. The performance advantage of the compensating zeros method is only realized when T presents a real constraint on the fixed-point filter design.

▲ 2) In the compensating zeros technique, the first cascaded section must be quantized so that it is different from the unquantized filter (i.e., $C_1'(z)$ must be different from $C_1(z)$). This affords the second cascaded section the opportunity to compensate for the quantization effects in the first section.

▲ 3) For nonlinear phase FIR filters the filter coefficients are no longer symmetric; consequently, (3) must be solved for both positive and negative frequencies to ensure that real coefficients are maintained.

▲ 4) The set of frequencies at which (3) is evaluated determines how well the compensated magnitude response matches the unquantized response. There is no optimal set of frequencies; instead, several frequency sets may need to be evaluated to identify the set that yields the closest match.

▲ 5) The original filter must be long enough so that when it is divided into the cascade structure, the sections have sufficient length so that compensation can occur.

▲ 6) It is desirable to quantize the shorter cascaded section first and then compensate with the longer cascaded section. The longer the compensating section, the larger the set of frequencies at which (3) is evaluated, and the better the match between the quantized and unquantized frequency responses.

▲ 7) The compensating zeros method can be employed for multiple cascaded sections. If a filter is divided into N cascaded sections, c_1 through c_N, then the design begins as described in the example for the first two cascaded sections, c_1 and c_2. Next, c_1 and c_2 are combined into one quantized filter, and c_3 is designed to compensate for the quantization effects in both $c_1$andc_2 [by solving (3)]. This process continues until the last cascaded section, c_N, is designed. Furthermore, not all cascaded sections have to be used to perform the quantization compensation; the choice is up to the designer.

Concluding Remarks

For hardware filter implementations that use the cascade multiplierless structure, the compensating zeros quantization method outperforms simple quantization given a small fixed value of T. The compensating zeros technique quantizes the cascaded sections so that the finite word-length effects in one section are guaranteed to compensate for the finite word-length effects in the other section. The algorithm involves no optimization—just the solution of a linear system of equations. Moreover, compensating zeros quantization ensures that: 1) the quantized filter's frequency response closely matches the unquantized filter's frequency response (magnitude and phase) and 2) the required hardware remains small and fast. This technique can be applied to any unquantized FIR filter, and it can exploit the cascade structure's ability to isolate some of the zeros in the filter.

Acknowledgments

This material is based upon work supported by the National Science Foundation under grants 0218672 and 0217894.

Kishore Kotteri received his B.S. in electronics engineering from the University of Mumbai in 1997. After graduation he worked for four years as a software consultant with TATA Consultancy Services. He is currently pursuing a Ph.D. in electrical engineering at Virginia Tech. His research interests are in the areas of digital signal processing, image processing, and communications.

Amy E. Bell is an assistant professor in the Department of Electrical and Computer Engineering at Virginia Tech. She received her Ph.D. in electrical engineering from the University of Michigan. She conducts research in wavelet image compression, embedded systems, and bioinformatics. She is the recipient of a 1999 NSF CAREER award and a 2002 NSF Information Technology Research award.

Joan E. Carletta is an assistant professor in the Department of Electrical and Computer Engineering at the University of Akron. She received a Ph.D. in computer engineering from Case Western Reserve University in 1995. Her research involves the design of digital hardware for applications that require intensive computation. Her work is funded by two NSF Information Technology Research awards.

References

[1] J.G. Proakis and D.G. Manolakis, *Digital Signal Processing Principles, Algorithms, and Application*, 3rd ed. Englewood Cliffs, NJ: Prentice Hall, 1995, pp. 500–652.

[2] A.V. Oppenheim, R.W. Schafer, and J.R. Buck, *Discrete-Time Signal Processing*, 2nd ed. Englewood Cliffs, NJ: Prentice Hall, 1999, pp. 366–510.

[3] C. Lim, R. Yang, D. Li, and J. Song, "Signed power-of-two term allocation scheme for the design of digital filters," *IEEE Trans. Circuits Syst. II*, vol. 46, no. 5, pp. 577–584, May 1999.

[4] T.800: Information Technology–JPEG2000 Image Coding System, ISO/IEC 15444-1:2002 [Online]. Available: http://www.itu.int

[5] I. Daubechies, *Ten Lectures on Wavelets*. Philadelphia, PA: SIAM, 1992, p. 277.