Determining Appropriate Precisions for Signals in Fixed-Point IIR Filters

Joan Carletta
The University of Akron
Electrical & Comp. Eng.
Akron, OH 44325-3904
+1 330 972-5993
carlett@uakron.edu

Robert Veillette
The University of Akron
Electrical & Comp. Eng.
Akron, OH 44325-3904
+1 330 972-5403
veillette@uakron.edu

Frederick Krach
The University of Akron
Electrical & Comp. Eng.
Akron, OH 44325-3904
+1 330 972-5993
fk@uakron.edu

Zhengwei Fang
The University of Akron
Electrical & Comp. Eng.
Akron, OH 44325-3904
+1 330 972-5993
zfang@uakron.edu

ABSTRACT

This paper presents an analytical framework for the implementation of digital infinite impulse response filters in fixed-point hardware on field programmable gate arrays. This analysis is necessary because FPGAs, unlike fixed register size digital signal processors, allow custom bit widths. Within the framework, the designer determines the number of bits necessary for representing the constant coefficients and the internal signals in the filter. The coefficient bit widths are determined by accounting for the sensitivity of the filter's pole and zero locations with respect to the coefficient perturbations. The internal signal bit widths are determined by calculating theoretical bounds on the ranges of the signals, and on the errors introduced by truncation in the fixed-point hardware. The bounds tell how many bits are required at any point in the computation in order to avoid overflow and guarantee a prescribed degree of accuracy in the filter output. The bounds form the basis for a methodology for the fixed-point digital filter implementation. The methodology is applied to the implementation of a second-order filter used as a compensator in a magnetic bearing control system.

Categories and Subject Descriptors

B.5.2 [**Register-Transfer Level Implementation**]: Design Aids – *automatic synthesis.*

General Terms

Design, Experimentation.

Keywords

Finite word length effects, infinite impulse response filter, design methodology, field programmable gate array.

This material is based upon work supported by the National Science Foundation under Grant No. 0113168.

DAC 2003, June 2-6, 2003, Anaheim, California, USA.

1 INTRODUCTION

Finite word-length effects are often a critical issue in the implementation of digital signal processing algorithms in finite precision hardware. This is particularly true for infinite impulse response (IIR) implementations with a high sampling rate, as the computational accuracy becomes more critical.

Field programmable gate arrays (FPGAs) are an ideal platform for implementation of IIR filters; using fixed-point structures and parallel computations, they provide a computational speed as much as two orders of magnitude greater than that possible with digital signal processors, and still offer the ability to reprogram [7]. With an FPGA, the designer can use a separate fixed-point format for each individual signal in a system; in contrast, on a digital signal processor, the register lengths are fixed, so the designer must work with a single fixed-point format. The ability to choose custom signal bit widths allows flexibility in overcoming numerical difficulties caused by finite word-length effects.

The design freedom inherent in FPGA implementation necessitates the use of a systematic algorithm for choosing coefficient and signal bit widths. This work outlines a practical methodology for choosing bit widths in IIR filters to guarantee a desired level of precision in the filter output. The coefficient precisions are chosen so as to limit perturbations in the transfer function poles and zeros. The signal bit widths are chosen both to accommodate the maximum signal amplitudes and to limit the maximum output error resulting from quantization. By the judicious use of variable bit widths, the methodology produces a high-quality implementation with less hardware than would be possible using a single fixed-point format for all the signals in the system.

Some past related work has focused on limit cycles and overflow, and their relationship to the bit widths in finite precision digital filters. Bounds on limit cycle amplitudes are derived in [5] and [2], giving guidance for the selection of the bit widths in fixed-point and floating-point filter implementations, respectively. It is shown in [6] that, if overflow does not cause instability for a theoretical infinite-precision system, then the quantization bounds of the computed signals can be chosen small enough that the overflow will likewise not cause instability for the finite-precision implementation. In contrast, the methodology presented in this paper addresses the issues of both overflow and limit cycle amplitude by the use of two different types of

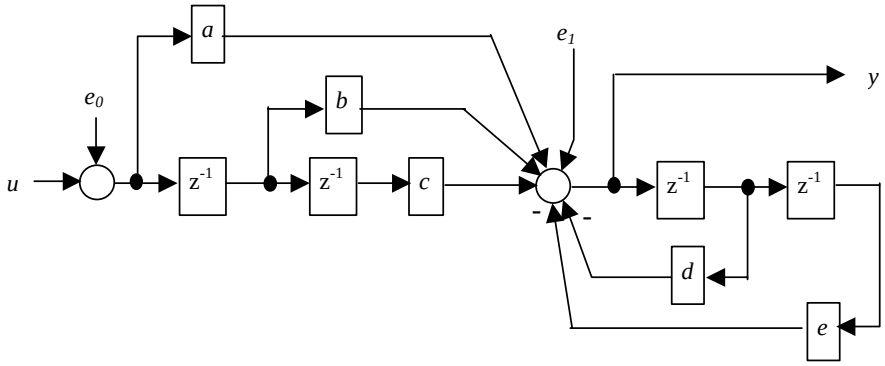


Figure 1. System simulation diagram for a direct-form realization of a second-order system. Truncation errors are represented as the fictitious inputs e_0 and e_1.

bounds. The first determines the maximum range of the calculated variables. It guarantees that, if a certain number of integer bits is used, overflow cannot occur in the calculations. The second determines the maximum amplitude of the error that can arise from truncation in calculating the filter output. It fixes the number of fraction bits that should be used to keep limit cycles within a given prescribed bound. Work in [3] is similar in that it presents a methodology for choosing bitwidths of signals, but it chooses the number of fraction bits so as to bound the variance of the output error, rather than the maximum amplitude of the output error.

The paper is organized as follows. Section 2 introduces a simple example filter and defines a computational structure for its implementation. Section 3 addresses how to quantize the filter coefficients, and Section 4 addresses how to choose an appropriate number of bits for calculated variables within the system. Section 5 summarizes the overall design methodology for infinite impulse response filters. In Section 6, the results of the methodology are illustrated by an example, which involves design of a second-order compensator used for the control of a magnetic bearing. Conclusions are drawn in Section 7.

2 COMPUTATIONAL STRUCTURE

It will be assumed that a digital IIR filter is given that has the desired properties when implemented in infinite precision. For purposes of illustration, we will consider a second-order filter, described by the *z*-domain transfer function

$$G(z) = \frac{Y(z)}{U(z)} = \frac{az^2 + bz + c}{z^2 + dz + e}, \qquad (1)$$

which represents the difference equation

$$y_{k+2} + d\,y_{k+1} + e\,y_k = au_{k+2} + bu_{k+1} + cu_k\,. \qquad (2)$$

A filter with this structure is used as a compensator in a magnetic bearing control system [7]. This compensator was derived by discretizing an analog control system for a specific sampling period. When implemented with a sampling period of 100 ns, appropriate values of the coefficients are:

$$a = 101.80 \qquad b = -203.40 \qquad c = 101.60$$
$$d = -1.968 \qquad e = 0.968 \qquad (3)$$

There are a number of possibilities for organizing the computations necessary to implement the discrete-time system. This paper focuses on the direct-form realization shown in Figure 1. This form clearly separates the hardware into a section for the zeros, on the left in the figure, and a section for the poles, on the right. It requires more registers, but fewer adders, than standard canonical realizations do. Figure 1 includes error "inputs" e_0 and e_1 to facilitate signal error bound computations in Section 4. These bounds determine the necessary bit widths for representing the filter coefficients and the stored variables. Similar bounds could be calculated for other realizations of the same filter. This would allow the comparison of the amount of hardware required for the various realizations to maintain a given level of filter performance. Such a comparative analysis is left for future consideration.

3 COEFFICIENT REPRESENTATION

In a fixed-point implementation, each controller coefficient *a* through *e* must be given a finite-precision representation. For each coefficient, both the number of integer bits and the number of fraction bits must be chosen. Choosing the number of integer bits is straightforward: a coefficient *c* requires $\lceil \log_2 c \rceil + 1$ bits to the left of the binary point. Choosing the number of fraction bits requires more care. In general, the coefficients must be approximated to fit in a finite-length register. The approximation can be interpreted as a perturbation of the coefficient from its ideal infinite precision value. The coefficients (3) place the poles of $G(z)$ at $\boldsymbol{z}$ = {0.9683, 0.9997}, and the zeros at $\boldsymbol{z}$ = {0.9981, 0.9999}. The poles and zeros of $G(z)$ are crowded together near $\boldsymbol{z}$=1, as is typical for discrete-time filters with short sampling periods. This makes the discrete-time calculations particularly sensitive to numerical errors. The truncation of the filter coefficients may even move a pole outside the unit circle, and therefore into the instability region. A careful analysis is required to ensure adequate precision of the coefficients.

When the denominator coefficients of the second-order filter are perturbed by Δd and Δe, respectively, the poles of the transfer function will also be perturbed by an amount [8]

$$\Delta p_1 = \frac{p_1 \Delta d + \Delta e}{p_2 - p_1} \qquad \Delta p_2 = \frac{p_2 \Delta d + \Delta e}{p_1 - p_2} \qquad (4)$$

where p_1 and p_2 denote the two poles. A similar relation holds for the perturbations of the zeros of the transfer function. The positions of the poles and zeros relative to the unit circle are critical to the performance of the filter; therefore, we require

$$\left|\frac{\Delta p_i}{1-|p_i|}\right| < \varepsilon\,, \qquad (5)$$

where ε is a prescribed maximum allowable percent change in pole location relative to the unit circle. Using equation (4), a conservative sufficient condition for (5) is

$$|\Delta d| + |\Delta e| < \varepsilon|1-|p_i|\,|\,|p_2 - p_1|\,, \quad i = 1,2. \qquad (6)$$

If d and e have the same number of fraction bits, then $\Delta d = \Delta e$, and (6) will hold if the number of fraction bits is given by

$$f = \left\lceil -\log_2\left(\varepsilon|1-|p_i|\,|\,|p_2 - p_1|\right)\right\rceil + 1\,, \quad i = 1,2. \qquad (7)$$

As an example, to guarantee ε = 10% for both poles**,** our compensator $G(z)$ requires 21 bits of fraction for coefficients d and e. Altogether (including integer bits), d and e would require 23 and 22 bits, respectively.

The above calculations are based on sufficient conditions (6), and therefore may yield conservative results. For this example, the condition (5) with ε = 10% is actually achieved using 21 bits for d and 18 bits for e. Similarly, a tolerance of ε = 10% for the zeros is achieved with bit widths of 25 bits, 26 bits, and 26 bits for coefficients a, b, and c, respectively. These are the coefficient bit widths that will be used in the sample FPGA-based filter implementation presented in the results section.

Given that the poles and zeros of the implemented system are sufficiently close to those of the infinite-precision system, we will now take the truncated coefficient values as the nominal ones. The range and the necessary precision of the calculated variables in the filter can be analyzed using the truncated values of the filter coefficients.

4 CALCULATED VARIABLE REPRESENTATION

For our purposes, a system consists of additions and multiplications that produce calculated variables, and registers that hold delayed versions of those variables. The next question in the fixed-point filter implementation is what fixed-point format to use for each calculated variable and register within the system. Here, we explicitly consider only those calculated variables that are the results of additions. Once the formats of the addition results are chosen, the formats for multipliers and registers can be derived.

Two types of bounds are used to determine appropriate representations for calculated variables. The first type gives the range of a calculated variable; this determines the maximum amplitude of the variable, and thus the number of integer bits needed to avoid overflow. The second type of bound relates truncation errors in the calculated variables to errors in the filter output. This type of bound tells how many fraction bits are required for calculated variables to ensure a desired precision in the filter output. The two types of bounds together determine the total bit width needed for the calculated variables. The bounds are determined by the analysis of various system transfer functions. The analysis can be done for either open-loop systems (appropriate for signal processing applications) or closed-loop systems (appropriate for control systems, where the filter is used as a compensator in a feedback loop). Open-loop analysis is presented in this paper.

4.1 Bounds on Range

For any calculated variable, the fixed-point representation must include enough integer bits to avoid the occurrence of overflow. To ensure this, we need to know the variable's range, or how large its magnitude can become.

Let $\boldsymbol{G}_{v,u}(z)$ represent the transfer function from the input of the system u to a particular variable v, and $g_{v,u}(k)$ the corresponding system impulse response. A bound [4] on the maximum value of $|v|$ is given by

$$\|v\|_\infty \le \|g_{v,u}\|_1 \cdot \|u\|_\infty\,, \qquad (8)$$

where $\|\cdot\|_\infty$ and $\|\cdot\|_1$ denote respectively the l_∞ and l_1 norms, defined by $\|x\|_\infty \equiv \sup_k\{|x(k)|\}$, and $\|x\|_1 \equiv \sum_{k=0}^{\infty}\{|x(k)|\}$. Note that both quantities on the right-hand side of the inequality (8) are well known. The l_1 norm of the impulse response can be computed numerically, and the l_∞ norm of the input signal is bounded by the range of the analog-to-digital (A/D) converter that supplies it.

As an example, consider the direct-form realization of the example second-order filter of Equation (1), depicted in Figure 1. This form has only one calculated variable, at the output of the adder; it happens to be the filter output y.

For the example system with a sampling period of 100 ns, the transfer function is

$$G_{y,u}(z) = G(z) = \frac{101.80z^2 - 203.40z + 101.60}{z^2 - 1.968z + 0.968}\,. \qquad (9)$$

The impulse response $g(k)$ of this system may be determined by the inverse z transform of the transfer function $G(z)$. Then $\|g\|_1$, the absolute sum of $g(k)$, may be computed. Here,

$$\|\boldsymbol{g}\|_1 = 197.96. \qquad (10)$$

Our A/D produces a ten-bit digital signal u that is the input of the system. For our particular application the analog signal to be processed lies in the voltage range ±4 V; therefore, any possible input signal to the filter satisfies $\|u\|_\infty \le 4$. Equation (8) tells us that $\|y\|_\infty \le 791.84$; that is, the magnitude of the output signal y can never exceed that value. As a result, eleven bits of integer should be used when calculating y, to allow a range of [–1024, +1024).

4.2 Bounds on Output Error

There are two main sources of error in the filter output y due to quantization for our system, namely

1. the error Δy_0 introduced by the limited resolution of the analog-to-digital (A/D) converter, and

2. the error Δy_{trunc} introduced by truncation in each summation of a calculated variable of the system.

Both types of errors may be represented as exogenous disturbance inputs to the system, with the filter calculations and input assumed to be otherwise ideal. For the direct-form realization of Figure 1, there are two places at which quantization error is introduced. These are shown on the simulation diagram as e_0 and e_1. The disturbance e_0 represents the A/D quantization error, and the disturbance e_1 represents the truncation error in the summation. Other realizations may have more than one summation; for a system with m summations, we use the notation e_i to represent a disturbance due to the truncation error in the summation for calculated variable v_i, for $i = 1, 2, \ldots m$.

If a particular disturbance e_i is the only source of error in the system, the output error is governed by the bound

$$\|\Delta y_i\|_\infty \le \|g_{y,e_i}\|_1 \cdot \|e_i\|_\infty \qquad (11)$$

where $\boldsymbol{g_{y,e_i}}(k)$ is the impulse response of the system from e_i to y. Here, $\|g_{y,e_i}\|_1$ is a constant, given the implementation of the filter, and $\|e_i\|_\infty$ is the maximum error introduced at the source of the quantization. Errors from multiple sources may be superimposed to find the total effect on the output y. The total output error is governed by the bound

$$\|\Delta y\|_\infty = \|\Delta y_0 + \Delta y_{trunc}\|_\infty \le \sum_{i=0}^{m} \|g_{y,e_i}\|_1 \cdot \|e_i\|_\infty . \qquad (12)$$

In order to design a filter whose output has a desired accuracy of at least ε, we must choose a precision Δ_i for each quantization error source i such that

$$\sum_{i=0}^{m} \|g_{y,e_i}\|_1 \cdot \Delta_i \le \varepsilon . \qquad (13)$$

Choosing a precision of Δ_i implies keeping $\lceil -\log_2(\Delta_i) \rceil$ fraction bits at that point in the calculation.

The transfer function from e_0 to y is the same as the transfer function from system input u to y. Therefore, working with the example for $T = 100$ ns,

$$G_{y,e_0}(z) = G(z) = \frac{101.80z^2 - 203.40z + 101.60}{z^2 - 1.968z + 0.968} . \qquad (14)$$

The l_1 norm of the impulse response of this transfer function is $\|g_{y,e_0}\|_1 = 197.96$. Our ten-bit A/D output is interpreted as three bits of integer (including sign) and seven bits of fraction. Since the A/D converter rounds its digital output to the nearest available quantized output, the worst case error is given by $\|e_0\|_\infty = 2^{-8}$ V. Using the error bound (11), the maximum error in y resulting from the A/D converter quantization is

$$\|g_{y,e_0}\|_1 \cdot \|e_0\|_\infty = 197.76 \cdot 2^{-8} = 0.773 \text{ V}. \qquad (15)$$

The only way to reduce the size of this error is to use an A/D converter that provides more bits of precision.

The second source of quantization error is the truncation of the lower order bits when calculating the sum; this is shown as e_1 on the simulation diagram of Figure 1. The corresponding transfer function (for $T = 100$ ns) is

$$G_{y,e_1}(z) = \frac{z^2}{z^2 + dz + e} = \frac{z^2}{z^2 - 1.968z + 0.968} . \qquad (16)$$

The l_1 norm of corresponding system impulse response is $\|g_{y,e_1}\|_1 = 97443$. The size of e_1 depends on the number of fractional bits that are retained in calculating the sum. Assuming that the result of the summation is truncated after f fraction bits, the maximum error can be $\|e_1\|_\infty = 2^{-f}$ V. Using the error bound (11), the maximum error in y due to truncation after summation is

$$\|\Delta y_1\|_\infty \le \|g_{y,e_1}\|_1 \cdot \|e_1\|_\infty = 97443 \cdot 2^{-f} \text{ V}. \qquad (17)$$

This error in y can be reduced by keeping more fraction bits.

Superimposing the effects of the two sources of quantization error, the total error in the filter output is governed by the bound

$$\|\Delta y\|_\infty \le 0.773 + 97{,}443 \cdot 2^{-f} \text{ V}, \qquad (18)$$

where f is the number of fraction bits retained in the calculation of the sum.

If the filter is to be used as a part of a closed-loop system, the analysis is similar, except that the impulse response functions $g_{y,e}(k)$ used for computing the bounds are those of the closed-loop system. Accordingly, the bounds will depend not only upon the filter itself, but also upon the system being controlled.

4.3 Bit Width Selection

Assuming that the number of bits provided by the A/D converter is given and fixed, the designer is left only with the choice of the number of fraction bits f in the sum calculation that produces the filter output y. Given that the constant term 0.773 V is already a significant fraction of the output range of ±8 V, it seems logical to choose f to ensure that the second term in the bound (12) is considerably smaller. We (rather arbitrarily) choose the condition that the second term should be no greater than 10% of the first, or

$$97443 \cdot 2^{-f} \le (10\%) \cdot 0.773. \qquad (19)$$

This condition implies $f \ge 20.2$; that is, 21 fraction bits are required.

Recall that the range of the computed variable has already been determined to require 11 integer bits. This means that, to guarantee the quantization errors produce a filter output error no greater than (110%)·0.773 = 0.850 V, a total of 32 bits are required to represent the calculated variable in the filter.

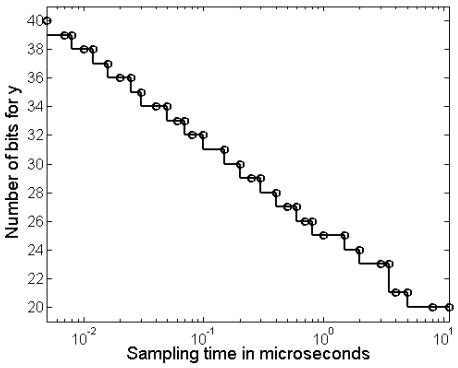


Figure 2. Bit-width chosen for calculated variable *y* using the proposed design methodology, as a function of sampling time.

The procedure for bit width selection was carried out for filter implementations with a wide range of sampling periods, using the same criteria for computational accuracy. The shorter the sampling period, the greater the range and error bounds, and therefore the more integer and fraction bits required to represent the digital variables. Figure 2 shows the total (integer plus fraction) number of bits required in the calculated variable as a function of the sampling period

5 FILTER IMPLEMENTATION METHODOLOGY

The range and error bounds can be used as the basis of a methodology for implementing fixed-point digital filters. The starting point is an infinite precision IIR filter with a desired response. The steps are:

1. Choose a computational structure for the filter computation. (Here, we have used a particular direct-form realization, but others are possible.)

2. Determine the necessary precision of the discrete-time filter coefficients so as to perturb the poles and zeros no more than a desired percentage ε relative to the unit circle, as described in Section 3.

3. Determine a bound on the filter output error that results from the A/D quantization as $\|\Delta y_0\|_\infty \le \|g\|_1 \cdot \Delta_0$, where $g(k)$ is the overall filter impulse response and Δ_0 is the quantization error of the A/D. Use the bound to identify an A/D bit width that will result in a tolerable output error. Keep in mind that some additional output error will be introduced by truncation in the filter.

4. Determine a bound on the maximum value of each calculated variable v_i that results from an addition operation as $\|v_i\|_\infty \le \|g_{v_i,u}\|_1 \cdot \|u\|_\infty$, where $g_{v_i,u}(k)$ is the filter impulse response from u to v_i, and $\|u\|_\infty$ is the full-scale value of the filter input. In order to avoid the possibility of overflow, represent the variable v_i using $\left\lceil \log_2\left(\|g_{v_i,u}\|_1 \cdot \|u\|_\infty \right) \right\rceil + 1$ integer bits, where the extra bit is for the sign.

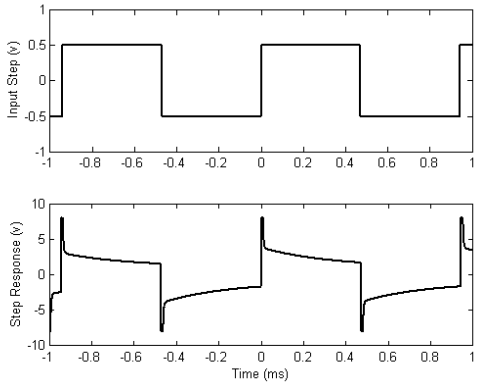
5. Define a fictitious input e_i perturbing each calculated variable v_i that results from an addition operation. For each calculated variable v_i, determine the corresponding norm $\|g_{y,e_i}\|_1$, where $g_{y,e_i}(k)$ is the impulse response of the system from e_i to the filter output y. Determine the number of fraction bits f_i for each calculated variable such that, for $\Delta_i = 2^{f_i}$, the bound on the output error resulting from truncations of all summations in the system, given by $\|\Delta y_{trunc}\|_\infty \le \sum_{i=1}^{m} \|g_{y,e_i}\|_1 \cdot \Delta_i$, is tolerably small. In our example, we have chosen the number of fraction bits to guarantee that this bound would be no more than 10% of that associated with the A/D quantization from Step 3.

6 RESULTS

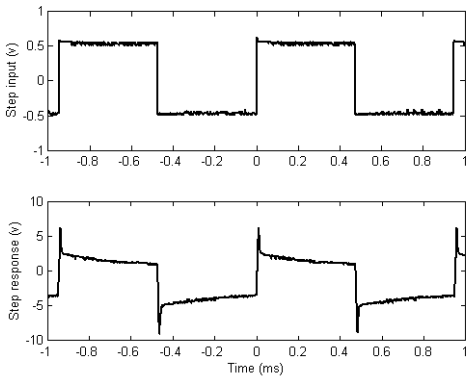
Two implementations of the second-order compensator with a 100 ns sample time were compared experimentally. The first uses the proposed methodology to choose appropriate bit widths. The second fixes the bit width of all internal variables of the system at 32 bits. Both compensators were implemented on Altera's DSP Development Board [1]. This board contains ten-bit analog-to-digital and digital-to-analog converters interfaced to an Altera Apex EP20K200EBC652-1X field programmable gate array. VHDL was written to program the hardware using Altera's Quartus II version 1.1 design package. The implementations were evaluated in both open- and closed-loop operation.

Table 1 compares the two systems. For each entry, the fixed-point data format is shown as $(n,-f)$, where n is the total number of bits and f is the number of fraction bits. Overall, the proposed methodology requires 14% fewer bits than the fixed-bit width system. This is reflected in the hardware cost; the proposed methodology requires 1404 logic elements, while the fixed bit version requires 4257 logic elements. Not all of this difference is due to the pure increase in number of bits; Altera's synthesis software used different multiplier structures for the larger 32-bit multiplications. For such large bit widths, larger but faster array multipliers are necessary to meet timing requirements.

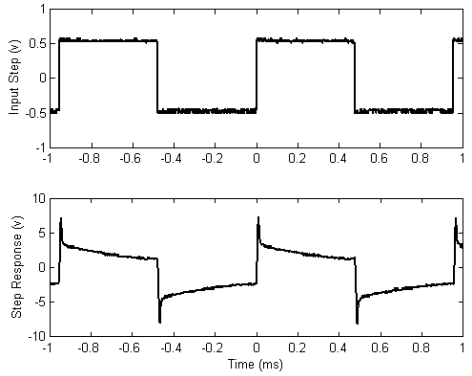
The filter implementations were evaluated in open-loop operation by looking at the response of the systems to a step input. Figure 3(a) shows an ideal, infinite-precision response of the compensator, while Figures 3(b) and (c) show the response of the fixed-point compensators designed using the proposed methodology and the fixed bit width approach, respectively. Both implementations have step responses close to the ideal, with similar noise levels. The implementations were also evaluated in closed-loop operation by using them to control the magnetic bearing for which the compensator was designed. Both fixed-point compensators successfully controlled the magnetic bearing system, with no noticeable



(a) using floating point precision in Matlab.



(b) for system with a 100 ns sampling time designed according to the methodology.



(c) for system with a 100 ns sampling time designed with a fixed bitwidth of 32.

Figure 3. Step responses of ideal and implemented systems.

difference. Therefore, the increased hardware expense of the fixed bit width version yields no benefit. Further, although based on worst case bounds, the methodology is not overly conservative; a compensator with two fewer bits for the variables was found to have significantly degraded open- and closed-loop responses, and was unable to control the magnetic bearing.

Quantity		using design methodology	using 32 bits Everywhere
Filter	a	(28, -20)	(32,-24)
	b	(29, -20)	(32,-23)
	c	(28, -20)	(32,-24)
	d	(21, -20)	(32,-31)
	e	(22, -20)	(32,-30)
Feedback y(k-1)		(32, -18)	(32,-18)
Feedback y(k-2)		(32, -18)	(32,-18)
Total Bits		192	224

Table 1. Coefficient and calculated variable bit widths.

7 CONCLUSIONS

A methodology for choosing customized bit widths for the coefficients and signals in a fixed-point digital filter implementation is developed. The methodology is applied to the implementation of a second-order IIR filter used as a compensator in a magnetic bearing control system. The bit widths chosen according to the given bounds produce a digital compensator that is similar in performance to a compensator designed using a more conservative, fixed bit width approach. Signal overflow is completely avoided, and the limit cycle amplitudes are kept below the desired level. The methodology is useful for managing the trade-off between computational accuracy and hardware utilization.

REFERENCES

[1] Altera Corporation, "APEX DSP Development Board (Starter Version)," Data Sheet, April 2002.

[2] P.H. Bauer, "Absolute response error bounds for floating point digital filters in state space representation," *IEEE Transactions on Circuits and Systems—II: Analog and Digital Signal Processing*, Vol. 42, No. 9, September 1995, pp. 610-613.

[3] G.A. Constantinides, P.Y.K. Cheung, and W. Luk, "The Multiple Wordlength Paradigm," *IEEE Symposium for Custom Computing Machines*, 2001.

[4] J. C. Doyle, B. A. Francis, and A. R. Tannenbaum, *Feedback Control Theory*, Macmillan, New York, 1992.

[5] B.D. Green and L.E. Turner, "New limit cycle bounds for digital filters," *IEEE Transactions on Circuits and Systems*, Vol. 35, No. 4, April 1988, pp. 365-374.

[6] K.K. Johnson and I.W. Sandberg, "A Separation Theorem for Finite Precision Digital Filters," *IEEE Transactions on Circuits and Systems--I: Fundamental Theory and Applications*, Vol. 42, No. 9, September 1995, pp. 541-545.

[7] F. W. Krach, B. P. Frackelton, J. E. Carletta, and R. J. Veillette, "FPGA-based implementation of digital control for a magnetic bearing," to appear in the *2003 American Control Conference*.

[8] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing: Principles, Algorithms, and Applications*, Prentice-Hall, New Jersey, 1996.