ACEPTED FROM OPEN CALL

A New Class of Floating-Point Data Formats with Applications to 16-Bit Digital-Signal Processing Systems

Manuel Richey, Honeywell International Inc.
Hossein Saiedian, University of Kansas

ABSTRACT

Sixteen-bit, programmable, digital-signal processors suffer from inadequate dynamic range and noise performance due in part to the use of standard data formats with few bits available for numeric precision. A solution to this problem is developed that involves the use of irregular data formats. A new class of irregular floating-point formats is developed. A specific format is derived from this class that provides greater dynamic range and improved noise performance for 16-bit DSP applications. An experiment with one of the new formats is conducted and analyzed, and improved performance is verified. The importance of our work and its potential applications are discussed.

INTRODUCTION

Writing software for a 16-bit digital-signal processing (DSP) application is difficult. One of the main reasons for this difficulty is that the data formats available on a standard 16-bit compiler and processor do not provide adequate dynamic range or noise performance for many DSP applications at high speed. This problem has been widely recognized, and proposed solutions have been developed and implemented to address it [1]. However, existing data-formatting solutions typically follow standard fixed-point and floating-point approaches. This article asserts that actual performance can be improved through the use of irregular data formats.

The fractional format is often used for implementing fixed-point DSP algorithms [2, p. 374]. In the embedded-C language [3], this format is complimented with the accum format for reducing round-off noise during multiply-accumulate operations. This pair of formats represents a substantial improvement over the C-int format but is often not sufficient for algorithms that require a large dynamic range.

Many floating-point formats exist to address the 16-bit dynamic-range issue, but the 16-bit binary versions suffer from poor noise performance. The floating-point format that typically is used is the 32-bit IEEE Standard 754 floating-point format [4]. In fact, this format is mandated for floating point in the C99 standard [5]. The IEEE 32-bit floating-point format is very comprehensive but potentially, very slow. If it is implemented in assembly language on a 16-bit DSP, arithmetic operations can take orders of magnitude longer to execute than equivalent fixed-point operations [6, p. 85], which are implemented directly in hardware.

The 16-bit DSP data-formatting problem has been around since engineers first began implementing DSP algorithms in FORTRAN. If we were to look at the data-formatting problem from outside of the existing paradigms and from an embedded DSP perspective, could we arrive at solutions that actually improve DSP algorithm performance? We say, "Yes."

A NEW CLASS OF FLOATING-POINT FORMATS

If we are to develop new data formats, we require a model from which to derive them. Therefore, we begin our development of new formats by first developing a class of data formats from which we can derive individual data formats. An appropriate class of data formats would allow specific formats to be derived that achieve similar noise performance to the current fixed-point fractional format, but with expanded dynamic range. Let's examine the fixed-point fractional format to determine why it has good noise performance.

Figure 1a plots the peak signal level vs. the peak round-off noise level for the largest 16-bit binary floating-point formats (including 16-bit fixed point). The notation used in this chart is as follows: s*M*e*N*, where s represents the sign bit, *M* represents the number of bits in the mantissa, e separates the exponent and mantissa, and *N* represents the number of bits in the exponent. The peak signal level vs. peak round-off noise level in decibels for each exponent is equal to $20 \times \log_{10}$((largest mantissa value)/(1/2 of smallest mantissa value)).

0163-6804/09/$25.00 © 2009 IEEE

This chart shows an advantage for the fixed-point fractional format only for the strongest signals. However, this advantage is real, and automatic gain-control (AGC) techniques can be used to take advantage of it. The other formats begin to outperform fixed-point fractional, as soon as a data value drops below one-fourth of the full range.

The ideal 16-bit format would achieve the performance of each of the formats in its area of strength. This implies a mantissa that is largest for the numbers closest to one and that gradually decreases in size as numbers get smaller. In Table 1, we can see this pattern is followed for the fixed-point sign/magnitude fractional format. We should point out that sign/magnitude is used in Table 1 for clarity, but two's complement is traditionally used in practice because it is easier to implement in hardware [7]. The problem with the fixed-point format is that precision falls off rapidly to zero. If we allow the precision to fall off at a slower rate, can we simultaneously achieve good noise performance and a wider dynamic range?

To accomplish this goal, we re-examine Table 1. From this table we discover a mechanism for recasting the 16-bit fixed-point format into a floating-point format. If the *X*s represent the mantissa and the mantissa is normalized, the number of leading zeros can represent an exponent (i.e., n leading zeros represents $2^{-(n+1)}$). The leading one becomes a mechanism for separating the exponent from the mantissa and is replaced with an implied one because the mantissa is normalized. This description provides the exact same formula as binary floating point, but the exponent is encoded differently. The equation becomes

$$value = (-1)S \times 1.M \times 2^{-E}$$

where S represents the number's sign, M represents the mantissa (the *X*s in Table 1), and E represents the exponent, which is equal to the number of leading zeros + 1. To drive this point home, let's represent the fixed-point number spectrum with the exponent at the right edge of the mantissa rather than at the left edge. This representation is shown in Table 2. This is very interesting, but so far all we have done is observe fixed point from a different perspective. We haven't gained anything.

Now, what would we have if we apply this exponent mechanism to both ends of the 16-bit word? This would give us two exponent fields, both of which are filled with zeros and separated from the mantissa by a one. The exponent from each field is given by the number of leading or trailing zeros. The available formats are rendered as shown in Table 3. This approach provides us with 120 exponents of varying degrees of precision (120 = summation of n for n = 1 to 15). This is an entirely different class of floating point than the standard binary floating point with fixed exponent length. In fact, a large number of specific irregular floating-point formats could be created from this floating-point class.

This class has several unique and important features that we utilize as we derive specific formats from it for DSP applications. These features are:

Variable precision: Unlike traditional binary

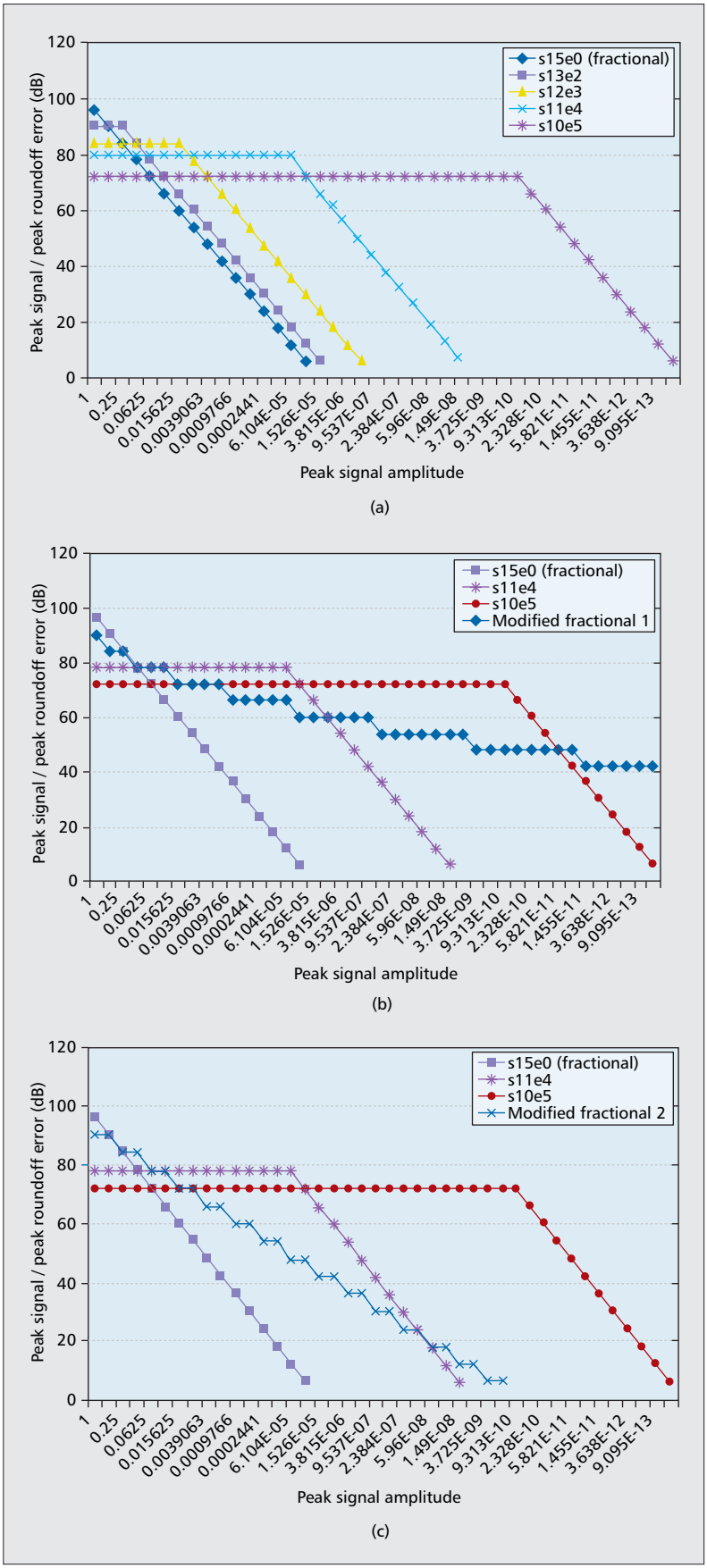

Figure 1. *Peak signal vs. peak round-off noise: a) for binary floating point formats; b) for the first fractional format; c) for the second fractional format.*

The main reasons for introducing a new fractional format are that this format has a path into a C data type with the embedded-C standard (ISO, 2003a), and it has a proven record for use in signal-processing applications.

Numeric range	Format S = sign bit X = either 1 or 0	Significant binary digits
1.0–0.5	S1XXXXXXXXXXXXXX	15
0.5–0.25	S01XXXXXXXXXXXXX	14
0.25–0.125	S001XXXXXXXXXXXX	13
0.125–0.0625	S0001XXXXXXXXXX	12
0.0625–0.03125	S00001XXXXXXXXX	11
0.03125–0.015625	S000001XXXXXXXX	10
0.015625–0.0078125	S0000001XXXXXXX	9
0.0078125–0.00390625	S00000001XXXXXX	8
0.00390625–0.001953125	S000000001XXXXX	7
0.001953125–0.0009765625	S0000000001XXXX	6
0.0009765625- 0.00048828125	S00000000001XXX	5
0.00048828125–0.000244140625	S000000000001XX	4
0.000244140625–0.0001220703125	S0000000000001X	3
0.0001220703125–0.00006103515625	S00000000000001	2
0.000030517578125	S000000000000001	1
0	S000000000000000	0

Table 1. *Fractional fixed point data in sign magnitude.*

floating-point formats, the mantissa length and numeric precision are variable for formats derived from this class. This is especially important for DSP formats.

Combining exponents: Individual exponents can be combined to produce exponents of higher precision. For example, two 12-bit exponents could be combined to form a 13-bit exponent. One 12-bit exponent could be combined with an exponent from each of the other precisions (11, 10, 9…1) to also create a 13-bit exponent.

Dual exponent mapping: A number's actual value for this class of formats is given by the equation: $value = (-1)S \times 1.M \times 2^{f(EL,ER)}$. The exponent is essentially determined by the number of contiguous zeros on each end of the mantissa. The mantissa itself consists of the bits in between the left-most and right-most ones (excluding the sign bit). These two ones can be used as separators because only zeroes are allowed in the exponent fields. Each individual binary exponent is derived by examining both the left-edge and right-edge exponents (*EL* and *ER*). A mechanism is then used (possibly, a table lookup) to map the two edge exponents into a single binary exponent $f(EL,ER)$.

Optimal sequences: Exponents can be ordered in any sequence, with the caveat that each binary range (e.g., 0.5 to 0.25) must be covered within the numeric range of the number format (e.g., 1.0 to 2^{-15} for fixed-point fractional). The dynamic range of the format is the ratio of the upper and of the lower limit (multiplied by two if rounding is employed).

We should point out that this class of floating-point formats is mainly applicable to problems that can benefit from an uneven distribution of precision. However, with the common use of AGC mechanisms, DSP applications fall into this category of problems. For applications that require an even distribution of precision across the numeric range, standard floating-point formats are more appropriately used.

This class of floating-point formats could easily be developed much further. But we don't require further development to derive from it data formats that are optimized for DSP. To derive a specific format from this class, we merely define the exponent function $f(EL,ER)$ for the individual derived format. With this class of formats as our primary tool, we now derive a data format optimized for 16-bit DSP applications.

A New Fractional Format

The authors have previously derived several data formats optimized for 16-bit DSP applications from this class of floating-point formats [8]. In this

Numeric range	Format S = sign bit M = mantissa 0 = exponent 1 = separator	Significant binary digits
1.0–0.5	SMMMMMMMMMMMMMM1	15
0.5–0.25	SMMMMMMMMMMMMM10	14
0.25–0.125	SMMMMMMMMMMMM100	13
0.125–0.0625	SMMMMMMMMMMM1000	12
0.0625–0.03125	SMMMMMMMMMM10000	11
0.03125–0.015625	SMMMMMMMMM100000	10
0.015625–0.0078125	SMMMMMMMM1000000	9
0.0078125–0.00390625	SMMMMMMM10000000	8
0.00390625–0.001953125	SMMMMMM100000000	7
0.001953125–0.0009765625	SMMMMM1000000000	6
0.0009765625- 0.00048828125	SMMMM10000000000	5
0.00048828125–0.000244140625	SMMM100000000000	4
0.000244140625–0.0001220703125	SMM1000000000000	3
0.0001220703125–0.00006103515625	SM10000000000000	2
0.000030517578125	s100000000000000	1
0	s000000000000000	0

This format provides almost double the dynamic range of standard fixed point at the cost of decreased precision at the very top of the numeric range (1.0–0.5). It also provides improved precision over fixed point throughout the remainder of the numeric range.

Table 2. *Fractional fixed point data format recast with trailing zeros.*

article we examine only one new data format, a new fractional format. The main reasons for introducing a new fractional format are that this format has a path into a C data type with the embedded-C standard (ISO, 2003a), and it has a proven record for use in signal-processing applications. If we can develop a fractional format with improved dynamic range that doesn't sacrifice noise performance, we will have accomplished something worthwhile.

Any format optimized for fractional DSP should take a clue from the noise performance of fixed point and have the largest mantissas at the top of the range. The mantissa should then fall off gradually to the smallest mantissa. Such a format derived from our new class is shown in Table 4 and illustrated in Fig. 1b. This format provides a significant dynamic-range improvement over the standard fixed-point format and the prevailing 16-bit floating-point format (s10e5 or binary16). It should also provide significantly improved noise performance over the binary16 floating-point format. Its only weakness is that a single bit of precision was sacrificed in the top two ranges when compared to fixed point. For a fractional format, this is significant, and we can do better by sacrificing some dynamic range.

By examining the binary fixed-point format, we learn that a single term from each of the ranges (A1 through M13 in Table 4) is combined to form a single 15-bit magnitude. This procedure is repeated to form a single 14-bit magnitude and on down the line. The result is that the binary fixed-point format can be considered a single instance of the new class of floating-point formats we developed in Table 3. To develop a new fractional format, we follow the same approach, but skip the creation of the first 15-bit magnitude. This leaves us with two terms of each exponent size, and we order them in decreasing precision as shown in Table 5 and illustrated in Fig. 1c.

This format provides almost double the dynamic range of standard fixed point at the cost of decreased precision at the very top of the numeric range (1.0–0.5). It also provides improved precision over fixed point throughout the remainder of the numeric range. As we discover, the increased dynamic range and additional overall precision actually improve the round-off noise performance of this format when compared to 16-bit fixed point.

Simulation and Results

A simulation was performed to validate improved noise performance for the second new fractional format. Here we provide a summary

Format S = sign bit M = mantissa 0 = exponent 1 = separator	Significant binary digits (assuming normalized Mantissa and implied leading 1)	Identifier (for later reference)
S1MMMMMMMMMMMM1	14	A1
S1MMMMMMMMMMMM10, S01MMMMMMMMMMM1	13	B1 B2
S1MMMMMMMMMMM100, S01MMMMMMMMMM10, S001MMMMMMMMMM1	12	C1 C2 C3
S1MMMMMMMMMM1000, S01MMMMMMMMM100, S001MMMMMMMMM10, S0001MMMMMMMMM1	11	D1 D2 D3 D4
5 values	10	E1–E5
…	…	
S1100…–S0…011	1	M1–M14
S1000…–S0…001	1	N1–N15
S000000000000000	0 & special use for S = 1	O1, O2

Table 3. *A new class of floating point formats.*

of the simulation and results, which are described in greater detail elsewhere [8].

An amplitude modulation (AM) receiver simulation was used to compare the noise performance for various data formats. The AM format was selected because the problem is well understood, and even if in decline, the format is still widely used. This simulation contained several typical DSP algorithms, which include the following: quantization to simulate 16-bit analog-to-digital (A/D) conversion, finite impulse response (FIR), and infinite impulse response (IIR) filters, demodulation, AGC, Hanning window, fast Fourier transform (FFT), and signal-to-noise ratio (SNR) measurement through Parseval's Theorem. AGC techniques were used following several stages to improve the native performance of the fractional format.

The following formats were simulated: IEEE 32-bit floating point (as a comparison baseline), s10e5 16-bit floating point, a 16-bit logarithmic format, fractional 16-bit fixed point, and the second new fractional format presented here. The simulation was performed for both weak-signal and strong-signal cases and both with and without the use of a single, large, post-multiply accumulator. Noise was not added to the simulation, so the resulting noise is a consequence of round-off errors during calculation, quantization to simulate A/D conversion, and out-of-band filter rejection (just over 50 dB). The simulation results are shown in Table 6.

As can be seen from Table 6, the new fractional format significantly outperformed the other 16-bit formats in terms of noise performance, and it approaches the performance of 32-bit floating point for this simulation. It also provides almost twice the dynamic range of traditional fixed point. The first fractional format was not simulated, but would drastically increase dynamic range with only a minor degradation of noise performance when compared to the second fractional format. It is still very useful for problems that require a larger dynamic range.

Better digital filters can also be achieved using the new fractional format than are possible using either fixed- or floating-point data types of equivalent word width. As shown in Fig. 2, a FIR-notch filter was created using the IEEE 32-bit floating-point format. The coefficients were then converted (with rounding) to the 16-bit fixed-point fractional format, the s10e5 16-bit floating-point format, and the 16-bit new fractional format. The coefficients were then converted back to floating point, and their magnitude response was plotted. As seen in Fig. 2, when the digital coefficients were expressed in the new fractional format, the rejection band improved by approximately 10 dB compared to the other two 16-bit formats. This improvement may become even more pronounced in IIR filters, which employ feedback.

Implementation Considerations

Having derived a data format that actually improves numeric performance for 16-bit DSP applications, we are left with several implementation issues that must be addressed before the format is truly useful. For non-real-time software applications such as simulation or data storage, the new fractional format has been coded into a C++ class with overloaded arithmetic operators that has provided excellent results. Unfortunately, this class performs arithmetic operations at a much slower rate than a traditional fractional format would. Furthermore, a data format with two exponent fields and a table-lookup mechanism may be difficult to implement in hardware and may execute more slowly than a traditional fractional format. The next logical step in our format development process is to render the second new fractional format into a form that is easily implemented in hardware. We accomplish this with a reorganization of the second new fractional format that allows easy encoding and decoding of the format into a 32-bit two's complement number.

Assume a 16-bit number partitioned into two fields. The first field is a single-bit field representing a shift flag. The second field is a 15-bit compressed two's complement number. The number of sign bits (leading ones or zeros) in the two's complement field is reduced by half (rounding up). This organization of bits proves equivalent to the new fractional format illustrated in Fig. 1c and defined in Table 5. Here we provide a mechanism for expanding the 15-bit field into a 32-bit two's complement number, and the shift bit indicates if the number is to be shifted to the left by one bit or not.

To decode the format, the leading digits of the 15-bit two's complement field that are all of the same binary value are to be doubled. In

Numeric range	Format S = sign bit M = mantissa 0 = exponent 1 = separator Red = actual digit Black = implied digit	*N*	*F*
1.0–0.5	S0.1MMMMMMMMMMMMM1	14	A1
0.5–0.25	S0.01MMMMMMMMMMMM1	13	B2
0.25–0.125	S0.001MMMMMMMMMMMM10	13	B1
0.125–0.0625	S0.00001MMMMMMMMMMM10	12	C2
0.0625–0.03125	S0.00001MMMMMMMMMMM100	12	C1
0.03125–0.015625	S0.000001MMMMMMMMMMM1	12	C3
0.015625–0.0078125	S0.0000001MMMMMMMMMM1	11	D4
0.0078125–0.00390625	S0.00000001MMMMMMMMMM1000	11	D1
0.00390625–0.001953125	S0.00000001MMMMMMMMMM100	11	D2
0.001953125–0.0009765625	S0.00000001MMMMMMMMMM10	11	D3
…	…	…	
	S0. "104 0s" 000000000000001	1	N15
0.0	0000000000000000	0	O1

Notes: *N* = number of significant binary digits. *F* = format from Table 3.

Table 4. *A first attempt at a new fractional format.*

Having derived a data format that actually improves numeric performance for 16-bit DSP applications, we are left with several implementation issues that must be addressed before the format is truly useful.

other words, if there are five ones or five zeros at the beginning of this field, you replace them with ten ones or ten zeros. If the shift bit is a zero, then you remove one of the leading sign bits at the front of the field. This number is then left justified into a 32-bit two's complement format, and the least significant bits are all set to zero. You now have expanded your 16-bit number into a 32-bit number with maximum precision for the very largest fractions.

The data can now be operated upon with a traditional 32-bit two's complement arithmetic unit. However, we should point out that a 15-bit DSP multiplier may be more appropriate because there are not more than 15 bits of precision in either multiplicand. The resulting product can be shifted into a 32-bit format after the 15×15 bit multiplication is complete.

Encoding from a 32-bit (or larger) two's complement word to a 16-bit new fractional format is accomplished by following the reverse operation. The number of leading zeros is halved (rounding up). The shift bit is then set to zero if the original number of leading sign bits is odd. Once the number of leading sign bits has been halved and the shift bit set, then rounding must be performed to compress the value into the resulting 15-bit two's complement format.

Although the number format itself has not increased in size (16-bits) with this approach, the computational element (multiplier, arithmetic logic unit [ALU], accumulator) may now have increased in size from 16 to 32 bits (at least for addition). This format should prove easy to implement in hardware but may result in a drop in throughput and an increase in power consumption due to the use of a 32-bit arithmetic unit together with encoding and decoding logic. Though this approach may slightly reduce the throughput of a 16-bit DSP device, it should not significantly affect the size or cost of the device because these attributes are driven primarily by the size of the main data and address buses. The use of a smaller 15×15-bit multiplier may also help with the size issue.

Some might argue that a hardware implementation of this new format would not be worthwhile because it does not directly improve important hardware performance parameters such as circuit complexity, delay, and power consumption. This point is granted. However, implementing this new format in hardware directly improves algorithm performance for many DSP applications. Obviously, not all DSP applications require additional dynamic range or improved noise performance, but other DSP applications go to great lengths algorithmically to obtain additional performance in these two categories. For these types of systems, improved algorithm

to be useful in a real-time application, the new format should be directly implemented in hardware, most conveniently in the arithmetic element of a programmable 16-bit DSP processor. Almost every application that uses a 16-bit DSP could be improved by using the new fractional format.

performance may indirectly improve power consumption, delay, algorithm complexity, cost, and size.

For many applications, the slight trade-off in throughput and power is well worth the improved dynamic range and noise performance achieved with the new format. Many applications that now require 32-bit floating point could be realized at less cost if this new data format were natively provided in a 16-bit programmable DSP device. Furthermore, many applications that currently utilize programmable 16-bit DSPs could experience improved performance with this new format.

Conclusions

In this article we introduced a new class of floating-point formats with uneven numeric precision. We illustrated how to derive specific formats from this class by creating two new fractional formats. We then presented simulation results to verify that the second new fractional format outperformed traditional 16-bit fixed- and floating-point formats in terms of noise performance and also dynamic range (for fixed point). Finally we remapped the new derived format into a form that is more convenient for hardware implementation.

Numeric range	Format S = sign bit M = mantissa 0 = exponent 1 = separator Red = actual digit Black = implied digit	*N*	*F*
1.0–0.5	S0.1MMMMMMMMMMMMM1	14	A1
0.5–0.25	S0.01MMMMMMMMMMMMM10	14	B1, C2, D3,..., M13
0.25–0.125	S0.001MMMMMMMMMMMM1	13	B2
0.125–0.0625	S0.0001MMMMMMMMMMMM100	13	C1, D2, ...
0.0625–0.03125	S0.00001MMMMMMMMMMM1	12	C3
0.03125–0.015625	S0.000001MMMMMMMMMMM1000	12	D1, E2, ...
0.015625–0.0078125	S0.0000001MMMMMMMMMM1	11	D4
0.0078125–0.00390625	S0.00000001MMMMMMMMMM10000	11	E1, F2, ...
0.00390625–0.001953125	S0.0000000001MMMMMMMMM1	10	E5
...	...	...	
	S0. "16 0s" 000000000000001	1	M13
0.0	0000000000000000	0	O1

Notes: *N* = number of significant binary digits. *F* = format from Table 3.

Table 5. *A second attempt at a new fractional format.*

Format	Weak signal SNR without accum	Weak signal SNR with accum	Strong signal SNR	Dynamic range
IEEE 754 32-bit floating point	31.97	31.97	50.53	1530
s10e5	8.44	7.90	42.06	252
16-bit logarithmic	8.23	8.32	38.61	385
16-bit fixed point fractional	13.30	24.93	44.42	96
New fractional (#2)	21.91	27.16	50.13	181

Note: Among the 16-bit formats blue indicates the best performer and red the second best performer.

Table 6. *Summary of simulation results (dB).*

We emphasized the fact that to be useful in a real-time application, the new format should be directly implemented in hardware, most conveniently in the arithmetic element of a programmable 16-bit DSP processor. Almost every application that uses a 16-bit DSP could be improved by using the new fractional format. Some obvious potential applications are:

- CD audio and other digital audio data formats
- Satellite TV and HDTV radio frequency (RF) data formats for signal processing
- Modems and other telephony applications
- Toys that employ speech generation or recognition
- Cell phone and software radio applications
- Implanted DSP devices (e.g., cochlear ear implants)

Some important aspects of this article that must be emphasized are:

- Both the new class of floating-point formats and the new fractional formats are applicable to data sizes other than 16-bits.
- The new class of formats may have applications outside the realm of DSP.
- Additional new data formats and more detailed development is provided in the authors' previous work [8].

For many years, the signal-processing industry has used primarily either fixed-point or binary floating-point number formats to represent digital signals. If the field of data formatting is opened up to include irregular formats, actual performance in terms of dynamic range and round-off noise can improve.

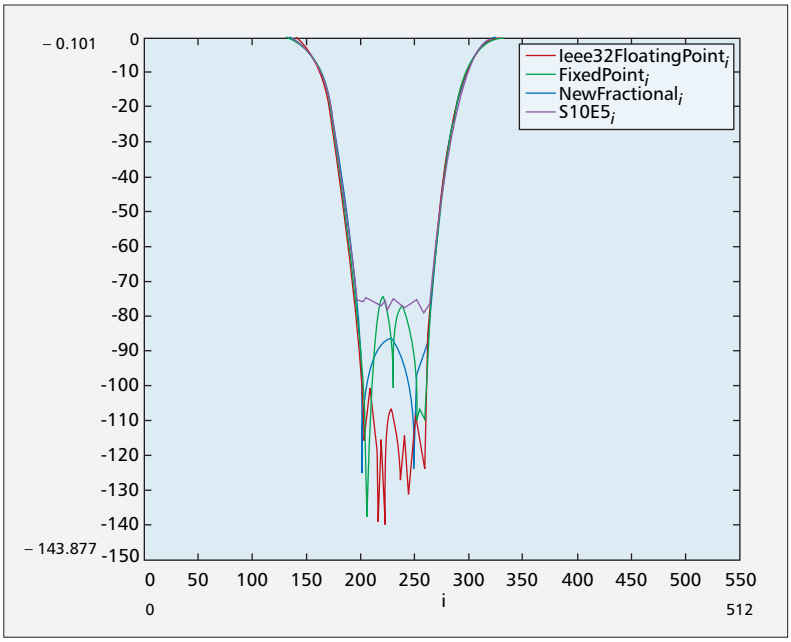

Figure 2. *A comparison of digital filter coefficient performance.*

References

[1] ISO/IEC JTC1/SC22 WG14/N854, "DSP-C: An Extension to IOS/IEC IS 9899:1990," 1998; http://www.open-std.org/jtc1/sc22/wg14/www/docs/n854.pdf (accessed Nov. 13, 2006).

[2] A. V. Oppenheim and R. W. Shafer, *Discrete-Time Signal Processing*, 2nd ed., Prentice-Hall, 1999.

[3] ISO/IEC JTC1/SC22 WG14/N1021, "Extensions for the Programming Language C to Support Embedded Processors: ISO/IEC DTR 18037," 2003; http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1021.pdf (accessed Nov. 13, 2006).

[4] IEEE Std P754 Draft 1.2.5, "Draft Standard for Floating-Point Arithmetic"; math.berkeley.edu/~scanon/754 (accessed Nov. 13, 2006).

[5] ISO, "Rationale for International Standard-Programming Languages — C," revision 5.1, 2003; http://www.open-std.org/jtc1/sc22/wg14/www/C99RationaleV5.10.pdf (accessed Oct. 18, 2006).

[6] S. W. Smith, *The Scientist and Engineer's Guide to Digital Signal Processing*, California Technical Publishing, 1999; http://www.analog.com/processors/learning/training/dsp_book_index.html (accessed Aug. 18, 2006).

[7] B. Parhami, *Computer Arithmetic: Algorithms and Hardware Designs*, Oxford Univ. Press, 1999.

[8] M. Richey, "The Application of Irregular Data Formats for Improved Performance in 16-bit Digital Signal Processing Systems," Master's thesis, Univ. of Kansas, Dec. 2006.

Biographies

HOSSEIN SAIEDIAN [SM] (saiedian@eecs.ku.edu) received his Ph.D. from Kansas State University in 1989. He is currently a professor and an associate chair in the Department of Electrical Engineering and Computer Science at the University of Kansas (KU) and a member of the KU Information and Telecommunication Technology Center (ITTC). His primary area of research is software engineering and information security. He has over 100 publications on a variety of topics in software engineering and computer science.

MANUEL RICHEY (manuel.richey@honeywell.com) graduated from the University of California at Davis in 1983 with a Bachelor's degree in electrical engineering. In 2007 he earned his Master's degree in computer science from the University of Kansas. He is a principal engineer at Honeywell International Inc. where he has worked since 1985. At Honeywell, he has worked primarily in the fields of digital signal processing and software radio. He holds seven U.S. patents and has four additional patents pending.