Fixed-Point vs. Floating-Point DSP for Superior Audio

- **Dynamic Range**
- **Proper Gain Setting**
- **Right Tool for the Right Job**

Greg Duckett
Terry Pennington
Rane Corporation

RaneNote 153
Revised 2005

Introduction

A popular belief is that for pro audio applications, floating-point DSPs are better than fixed-point DSPs – it depends, as you will learn. Read on to see why you may be paying too much and receiving too little when it comes to buying DSP signal processors.

A lot of confusion abounds in the audio industry over the issue of DSP internals. Things like: "Our box uses 32-bit floating point DSP chips, while their box uses only 24-bit fixed-point DSP chips. Obviously 32 is better than 24." Obviously – if you don't know the facts. What they fail to point out is the "rest of the story," which is that the 24-bit fixed-point DSP box can operate in "double-precision" mode, making it a 48-bit box. And in this case, 48 really is better than 32, only it has little to do with size. With today's technology, both fixed-point and floating-point can be equal if chosen correctly.

Since fixed-point is the most picked on, let's begin with how it can be superior to floating-point. Here is the executive summary of why fixed-point DSPs can make for superior audio:

1. Less dynamic range (yes, in DSPs used for audio, this can be a feature).
2. Double-precision capable 48-bit processing.
3. Programming flexibility that can guarantee proper behavior under the adverse conditions presented by audio signals — truly one of nature's oddest phenomena.
4. Lower-power consumption (floating point hardware is more complicated than fixed point; more transistors require more watts).

With that said, let's back up and review a few things.

A truly objective comparison of DSP architectures is not easy; in fact, it may not be possible. *In the end, it is the application and the skill of the software programmer implementing the application that determines superior audio performance.* But people don't want to hear that. They want the easy answer. Everyone is looking for the secret word, the single number that defines the difference and makes the duck drop down and deliver a $100 bill (*apologies made to all readers who have not seen the original* You Bet Your Life*, NBC 1950-1961, TV shows hosted by Groucho Marx*). Yet the truth is that there is no single number that quantifies the differences. Not the number of bits, the MIPS or FLOPS rate, the clock rate, the architecture, or any one thing.

Two distinct types of DSP implementations dominate pro audio applications: one utilizes *fixed-point* processing while the other features a *floating-point* solution. Both produce the same results under most conditions; however, it is the word "most" that creates the difference.

Looking under the hood of an IEEE 32-bit floating-point processor and a 24-bit fixed-point processor reveals that each DSP design offers the same 24-bit processing precision — *precision* is not the issue. The defining difference is that the fixed-point implementation offers double precision, while the floating-point device features increased dynamic range. In floating-point processors scaling the data increases dynamic range, but scaling does not improve precision, and in fact degrades performance for audio applications (more on this later). And it turns out that the strength of the fixed-point approach is the weakness of the floating-point, giving fixed-point a double advantage.

The benefit is most obvious in low frequency audio processing. This is important since most of the energy in audio lies in the low-frequency bands (*music and speech have an approximate 1/f spectrum characteristic, i.e., each doubling of frequency results in a halving of amplitude*). The simple truth is that the floating-point technique struggles with large amplitude, low-frequency computations. In fact, building a high-Q, low frequency digital filter is difficult no matter what method you use, but all things considered fixed-point double-precision is superior to floating-point single-precision.

To thoroughly explain, in a scientific engineering manner, the advantages and disadvantages of the two techniques as they relate to broadband audio applications is a vastly complex subject and lies beyond the scope of this article. An objective direct comparison involves a steep slippery slope of complexities, qualifications and definitions, all necessary in order to avoid apples to oranges error. A task as daunting as this has already been done by Dr. Andy Moorer[1] (cofounder of the Stanford University Center for Computer Research in Music and Acoustics, then CTO at the Lucasfilm Droid Works, then cofounder of Sonic Solutions, and now Senior Computer Scientist at Adobe Systems) and is recommended for the detail-curious and mathematically courageous. The goal here is to draw a simplified illustration of the audio challenges faced by each DSP solution. (See the *Let's Be Precise About This…* section for a more in-depth mathematically oriented example — sorry, but mathematics is all we are dealing with here.)

Dynamic Range

Higher dynamic range is better, yes? Just as lower distortion is better and lower noise is better. We're reminded of a sales guide published by a hi-fi manufacturer in the mid-seventies: This guide had a "lower is better" and "higher is better" approach to equipment specifications. The author of that promotional material would be shocked to hear an audio manufacturer claim that higher dynamic range can be a problem. Nonetheless, it is a fact when examined in relationship to the ultra-high dynamic range capabilities of the 32-bit floating-point processors found in some of the current DSP audio signal processors.

As mentioned earlier, both DSP designs have a 24-bit processor for the mainstream functions. The fixed-point technique adds double precision giving it 48-bit processing power, while the floating-point design adds an 8-bit exponent. The 8-bit exponent gives the floating-point architecture an astonishing dynamic range spec of 1500 dB (8-bits = 256, and 2^{256} equals approximately 1500 dB) which is used to manipulate an operating window, within which its 24-bit brain operates. Floating-point processors automatically scale the data to keep it within optimum range. This is where the trouble lies and this is why fixed-point is better than floating-point for audio. It is not that the dynamic range is the problem so much as the automatic scaling over a 1500 dB range that is the problem. Fixed-point, with its 48-bits, gives you 288 dB of dynamic range – enough for superior audio – but the programmer has to scale the data carefully. Floating-point programmers leave it up to the chip, but, unless they are careful, that creates serious errors and noise artifacts. All the jumping about done by the continuous signal boosting and attenuating can produce annoying noise pumping.

Floating-point evangelists flaunt their DSP dynamic range as a plus, but it only shows their weakness. *The dynamic range specification of a DSP chip has little to do with the overall dynamic range of the finished product.* The dynamic range of the "box" is bounded by the A/D converter on its input, to some extent on the processing in the center of the device, of which the DSP chip is a part, and on the D/A converter on the output (*even without converters the output of both DSP types is a 24-bit fixed-point word*). The dynamic range of a DSP chip is the ratio between the largest and smallest numbers it can handle. If a DSP device sporting an advertised dynamic range of 1500 dB resides between the input converters and the output converters its contribution to the overall dynamic range of the product is limited to the dynamic range of the converters. Is this a bad thing? No, not in itself.

What's bad about a floating-point processor with a dynamic range of 1500 dB is that it scales its processing range based on the amplitude of the signal its dealing with, but when dealing with signals of differing amplitudes (i.e., real audio), the scaling may not be optimized for the mixed result. When dealing with audio signals the installer cannot simply ignore the subtleties of system setup because they have a floating-point processor in their box.

Consider the typical audio mixer scenario: At any given moment a mixer can have multiple levels present at its many input ports. Input-1 might have a high-level sample to deal with while Input-2 has a very low level, Input-3 somewhere in the middle of its upper and lower limits and so on. A 32-bit floating-point DSP chip makes a determination about the appropriate window within which to work on a sample-by-sample basis but finally represents its calculations in the same 24-bit manner as its fixed-point counterpart. Even in a simple two-channel stereo processor signal levels between channels, while similar in average level, can be vastly different instantaneously due to phase differences.

Nothing is gained by using a floating-point device in an audio application but much may be lost. It does not have the 48-bit double precision capability of a fixed-point solution, and noisy artifacts may be added.

Importance of Proper Gain Setting

The reality here is that so long as we have finite precision/dynamic range in the converters and DSPs, the installer plays the final and most important role in maintaining the proper processing window alignment for a given installation. Improperly set gain structure can overload fixed-point processors. While floating-point DSPs give the flexibility to misadjust the system (too much internal gain) without noticeable internal clipping, they

still suffer the unintended consequences of the misalignment (say, in trying to mix two channels of very different audio levels) that floating-point processors cannot fix. They merely mask the problem from the installer's view. Or, worse, produce audible and annoying rise in quantization noise when filters are used below 100 Hz. In this sense the fixed-point processors force the installer to maintain the 144 dB processing window by avoiding internal clipping through proper gain structure/setup and so make maintaining overall quality easier than floating-proper processor based boxes.

Double Precision

The double precision 48-bit processing is used when long time constants are required. This occurs when low frequency filters are on the job and when compressors, expanders and limiters are used with their relatively slow attack and release times. If 24 bits are all that are available when more precision is required, the results are a problem. The function misbehaves and the least damaging result is poor sound quality. The worst result is amplifier or loudspeaker damage due to a misbehaving DSP crossover, making double precision a must-have for superior audio.

Examples and Counterexamples

Floating-point evangelists like to use an example where the processor is set up for 60 dB attenuation on the input and 60 dB make-up gain on the output. Leaving aside the absurdity of this fabricated example, let's use it to make our fixed-point-is-better point: add a second input to this example, with the gain set for unity, a 0 dBu signal coming in, and configure the processor to sum both these channels into the output and listen to the results — you will not like what you hear.

Another revealing example is how you never hear floating-point advocates talk about low-frequency/high-Q filter behavior. The next time you get the opportunity, set up a floating-point box parametric filter for use as a notch filter with a center frequency of 50 Hz and a Q of 20. First listen to the increase in output noise. Now run an input sweep from 20 Hz to 100 Hz and listen to all the unappetizing sounds that result. Audio filters below about 100 Hz require simultaneous processing of large numbers and small numbers — something fixed-point DSPs do much better than their floating-point cousins.

Free the Developers

The real determinant of quality in audio DSP is the skill of the programmers. They must devise accurate and efficient algorithms; the better their understanding of the (sometimes-arcane) math, the better the algorithm; the better the algorithm, the better the results. Fixed-point processing delivers a load of responsibility to the hands of the developer. It also delivers an equal amount of flexibility. A talented engineer with a good grasp of exactly what is required of a DSP product can fashion every detail of a given function down to the last bit. This is not so with floating-point designs. They offer an ease of programming that is seductive, making them popular when engineering talent is limited, but not the best choice. On one hand it is easier to program but on the other hand it is less controlled as to the final results — and, as we all know, that is what is important.

The Right Tool for the Right Job

If fixed-point DSP devices are so good, then why do floating-point DSPs exist? Fair enough question. They exist because DSP applications differ widely. Some of the more popular floating-point applications are found in physics, chemistry, meteorology, fluid dynamics, image recognition, earthquake modeling, number theory, crash simulation, weather modeling, and 3-D graphics. If you are designing an image processor, a radar processor, anything to do with astronomy, or a mathematics matrix inverter, the choice is clearly a floating-point solution. As always, the application dictates the solution.

What is required in a floating-point DSP to achieve superior audio? Here are some pretty nasty "ifs" necessary for floating-point to overtake fixed-point: *if* it is a 56-bit floating-point processor (i.e., 48-bit mantissa plus 8-bit exponent) or 32-bit with double-precision (requiring a large accumulator), *if* the parts run at the same speed as the equivalent fixed-point part, *if* they use the same power, and *if* they cost the same, then the choice is made.

Another possibility is if the floating point DSPs evolve to offer significantly more processing power for the same price (enough to overcome the low-frequency, high-Q issues in firmware) and offer a compatible peripheral chip set, then this could tip the scales even if they still offer only a 32-bit fixed numerical format.

Let's Be Precise About This ...

An example is the best way to explain how you lose precision when floating-point processors scale data. Assume you have two mythical 3-digit radix-10 (i.e., decimal) processors. One is "fixed-point", and one is "floating-point." For simplicity, this example uses only positive whole numbers. (On real fixed- or floating-point processors, the numbers are usually scaled to be between 0 and 1.)

The largest number represented in single precision on the fixed-point processor is 999. Calculations that produce numbers larger than 999 require double precision. This allows numbers up to 999999.

Let the floating-point processor use 2 digits for the exponent, making it a 5-digit processor. This means it has a dynamic range of 0 to 999×10^{99} = HUGE number. To see how this sometimes is a problem, begin with the exponent = 0. This allows the floating-point processor only to represent numbers up to 999 – same as the fixed-point single-precision design. Calculations that produce numbers larger than 999 require increasing the exponent from 0 to 1. This allows numbers up to 9990. *However*, notice that the smallest number (greater than zero) that can be represented is $1 \times 10^1 = 10$, *meaning numbers between 1-9 cannot be represented (nor 11-19, 21-29, 31-39, etc.).* Increasing the exponent to 3 only makes matters worst, but you can cover (almost) the same range as the fixed point processor (up to 999000); however the smallest number now represented is $1 \times 10^3 = 1000$, *meaning numbers between 1 and 999 cannot be represented*. And the next increment is $2 \times 10^3 = 2000$, meaning the represented number jumps from 1000 to 2000. So that now numbers between 1001 to 1999 cannot be represented. With exponent = 3, each increment in the mantissa of 1 results in an increase in the number of 1000, and another 999 values that cannot be represented.

Is this as big a problem as it first appears – well, yes and no. At first it looks like the floating-point processor has lost the ability to represent small numbers for the entire calculation's time, but the scaling happens on a *per-sample* basis. The loss of precision only occurs for the individual samples with magnitude greater than 999. Now you might think that everything is fine, because the number is big and it does not need the values around zero. But a few wrinkles cause trouble. When calculations involve large and small numbers *at the same time*, the loss of precision affects the small number and the result. This is especially important in low-frequency filters or other calculations with long time constants. Another wrinkle is that this happens automatically and beyond the control of the programmer. If the programmer does not employ the right amount of foresight, it could happen at a bad time with audible results.

In the fixed-point case, the programmer must explicitly change to double precision – there is nothing automatic about it. The programmer changes to double precision at the start of the program section requiring it and stays there till the work is done.

The Big and the Small

Over and over in audio DSP processing you run into the same simple arithmetic repeated over and over: multiply one number by another number and add the result to a third number. Often the result of this multiply-and-add is the starting point for the next calculation, so it forms a running total, or an accumulation, of all the results over time. Naturally enough, adding the next sample to the previous result is called an "accumulate" and it follows that a multiply followed by an accumulate is called a MAC. MAC's are the most common of all operations performed in audio DSP, and DSP processors typically have special hardware that performs a MAC very, very quickly.

As results accumulate, errors also accumulate. As well, the total can get large compared to the next sample. To show this in action return to the mythical 3-digit processors. Say we have the series of numbers shown in the row labeled "Samples" in Table 1; a strange looking set of numbers, perhaps, but it represents the first part of a simple sine wave. Multiply the first number by a small constant (say, 0.9) and add the result to the second number: 0 x 0.9 + 799 = 799. Multiply this result by 0.9 and add it to the third number: 799 x 0.9 + 1589 = 2308. And again: 2308 x 0.9 + 2364 = 4441. Continue this pattern and it forms a simple digital filter. The results using double precision fixed-point are shown in the row labeled "Fixed-Point Results" in Table 1.

Sample #	Samples	Fixed-Point Results	Floating-Point Results
1	0	0	0
2	799	799	799
3	1589	2308	2290
4	2364	4441	4420
5	3115	7112	7080
6	3835	10236	10200
7	4517	13729	13600
8	5154	17510	17300
9	5739	21498	21200

Table 1: Results Between Floating- and Fixed-Point Accumulation

What about the floating-point processor? Start with exponent = 0. The results are: 0, 799, … the next number is too big, so increase the exponent to 1 … 2290, 4420, etc. Notice that the floating-point values are smaller than they should be because the limited precision forces the last one or two digits to be 0. It's easy to see that each result has an error, and the errors are carried forward and accumulate in the results. Algorithms with long time constants, such as low frequency filters, are especially prone to these errors.

You'll also notice that the accumulated values are getting larger than the input samples. The long time constant in low frequency filters means that the accumulation happens over a longer time and the accumulated value stays large for a longer time. Whenever the input signal is near zero (at least once every cycle in a typical audio signal) the samples can be small enough that they are lost; because the accumulated value is large, the samples fall entirely outside the precision range of the floating point processor and are interpreted as zero. The double precision available in the fixed point processor helps the programmer to avoid these problems.

Further Study

Digital audio is a vast and complex subject with many subtleties when it comes to superior signal processing. An article this short touches only some of the important issues. Selecting just one book from all the possibilities for recommendation for further study is easier than it may seem. That book is John Watkinson's *The Art of Digital Audio, 3rd ed.* (Focal Press ISBN 0-240-51587-0, Oxford, England, 2001). The title says it all. One of the best digital audio references you can own.

References

1. Moorer, James A. (www.jamminpower.com) "48-Bit Integer Processing Beats 32-Bit Floating Point For Professional Audio Applications," presented at the 107th Convention of the Audio Engineering Society, New York, September 24-27, 1999, preprint no. 5038.