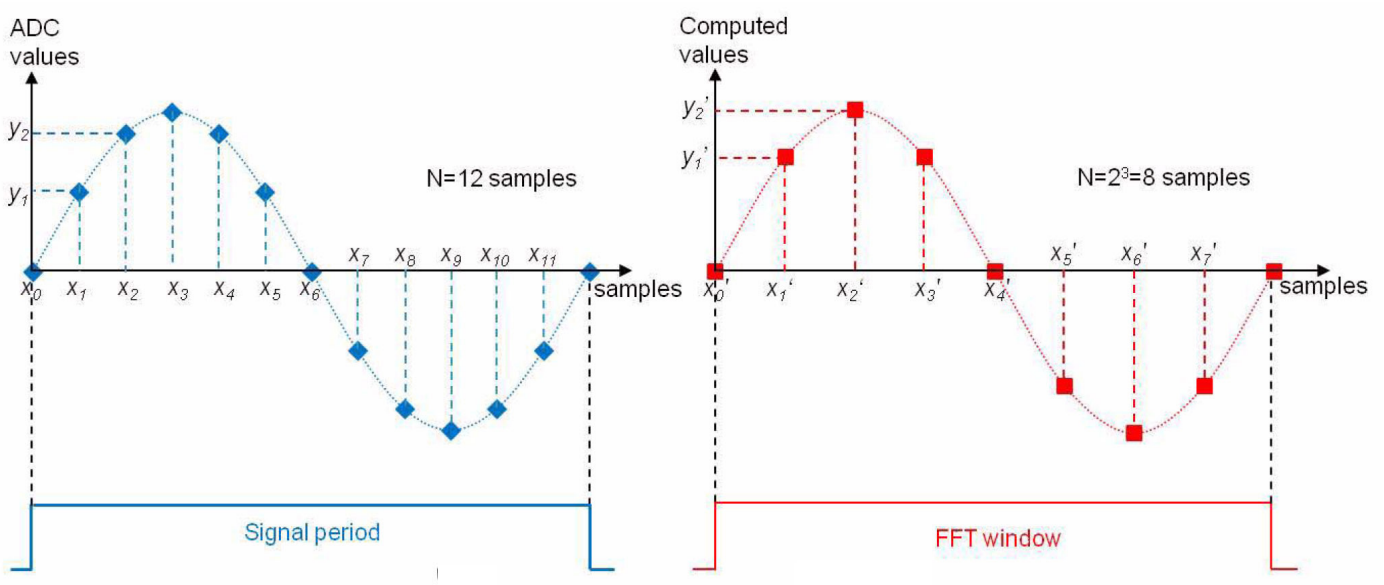
VIVIDSPARKS
CASE STUDY

Reversible FFTs Using POSIT™ Arithmetic



POSIT 16 bit has shown that higher accuracy and larger dynamic range, can perform FFTs so accurately that a forward inverse FFT restores the original signal perfectly. "Reversible" FFTs with POSITs are lossless, eliminating the need for 32-bit or higher precision.

Dr. John Gustfason,
National Supercomputing Center (NSCC),
Singapore

Introduction

The computation of the Discrete Fourier Transform (DFT) using the Fast Fourier Transform (FFT) algorithm has become one of the most important and powerful tools of High Performance Computing (HPC). FFTs are investigated to demonstrate the speed and accuracy of POSIT arithmetic when compared to floating-point arithmetic. Improving FFTs can potentially improve the performance of HPC applications such as CP2K, SpecFEM3D, and WRF, leading to higher speed and improved accuracy. “Precision” here refers to the number of bits in a number format, and “accuracy” refers to the correctness of an answer, measured in the number of correct decimals or correct bits. POSIT arithmetic achieves orders of magnitude smaller rounding errors when compared to floats that have the same precision. Thus, the commonly-used 32-bit and 64-bit precision floats can potentially be replaced with 16-bit and 32-bit posits respectively in some applications, doubling the speed of communication-bound computations and halving the storage and power costs.

Challenges

The Fast Fourier Transform (FFT) is required for chemistry, weather, defense, and signal processing for seismic exploration and radio astronomy. It is communication-bound, making supercomputers thousands of times slower at FFTs then at dense linear algebra. The key to accelerating FFTs is to minimize bits per datum without sacrificing accuracy and also reduce storage and power costs. The 16-bit fixed point and IEEE float type lack sufficient accuracy for 1024 and 4096-point FFTs of data from analog-to-digital converters.

Approach

To test the effectiveness of POSITs, 1024-point FFTs are used as the sizes most commonly found in the literature. Both radix-2 and radix-4 methods are studied here, but not split-radix. Although modified split-radix has the smallest operation count (about $3.88N \log_2 N$), fixed point studies show it has poorer accuracy than radix-2 and radix-4 methods.

Both Decimation-In-Time (DIT) and Decimation-In-Frequency (DIF) methods are tested. The DIF approach introduces multiplicative rounding early in the processing. Intuition says this might pollute the later computations more than the DIT method where the first passes perform no multiplications. Empirical tests are conducted to check this intuition with two numerical approaches:

- 16-bit IEEE standard floats, with maximum use of fused multiply-add operations to reduce rounding error in the multiplications of complex numbers,

- 16-bit POSITs (es = 1) with exactly the same operations as used for floats

For each of 24 combinations of data-point size, radix, decimation type, and numerical approach, random uniform distribution input signals in the range [-1; 1) at the resolution of a 12-bit ADC are created. The 12-bit fixed-point ADC inputs are first transformed into their corresponding 16-bit POSITs and floats as shown in Figure.1. A “round-trip” (forward followed by inverse) FFT is then applied before the results are converted (with rounding) back to the 12-bit ADC fixed point format (represented by ADC’ in Figure. 1). If no errors and rounding occur, the original signal is recovered perfectly, i.e. ADC = ADC’.

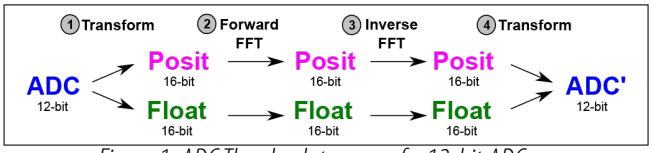


Figure 1: ADC The absolute error of a 12-bit ADC.

The absolute error of 12-bit ADC input can thus be computed as shown in Equation 1. This error represents the rounding errors incurred by POSITs and floats respectively.

$$absolute\ error = |ADC' - ADC| \qquad 1$$

To evaluate the accuracy of POSITs and floats, the vector of absolute errors of all ADC inputs is evaluated. Three flavors of measures, the maximum (L_∞ norm), RMS (L_2 norm) and average (L_1 norm) of the vector are computed.

To gain additional insights to the error, the units in the last place (ULPs) metric is used. ULP error measure is superior to relative error for measuring pure rounding error. ULP error can be computed as follows as shown in Equation 2. For a 12-bit fixed point ADC, *ulp(ADC')* is a constant value (2^{-11} for input in [-1; 1) range).

$$ULP\ error = \frac{ADC' - ADC}{ulp(ADC')} \qquad 2$$

Where *ulp(ADC')* is one unit value of ADC' the last place unit.

In the case of a POSITs and floats, even if an answer is correctly rounded, there will be an error of as much as 0.5 ULP. With each additional arithmetic operation, the ULP errors accumulate. Consequently, to minimize the effect of rounding errors, it is important to minimize the ULP error.

The IEEE standard also defines a multitude of rounding modes, from round-to-nearest (two forms: ties-round-to-nearest-even, and ties-round-away-from-zero), round-down and round-up, to round-towards-zero (= truncation). These alternative rounding modes are useful in diagnostic numerical instability but add a tremendous amount of hardware complexity for the uncommon use case.

For 16-bit fixed point, we rely on analysis because it obviates experimentation. After every pass of an FFT, the FFT values must be scaled by 1/2 to guarantee there is no overflow. For an FFT with 2^{2k} points, the result of the 2k scalings will be an answer that is too small by a factor of 2^k, so it must be scaled up by a factor of 2k, shifting left by k bits. This introduces zeros on the right that reveal the loss of accuracy of the fixed point approach. For a 1024-point FFT, the loss is 5 bits of accuracy. In FPGA development, it is possible to use non-power-of-two data sizes easily, and fixed point can be made to yield acceptable 1024-point FFT results if the data points have 18 bits of precision. Since fixed point requires much less hardware than floats (or POSITs), this is an excellent approach for special-purpose FPGA designs. In the more general computing environment where data sizes are power-of-two bits in size, a programmer using fixed point format has little choice but to upsize all the values to 32-bit size. The same will be shown true for 16-bit floats, which cannot achieve acceptable accuracy.

Benefits

Figure. 2 shows the average RMS errors for 1024-point tests, representing hundreds of thousands of experimental data points. The vertical axis represents Units in the Last Place (ULP) at the resolution of a 12-bit ADC as described in the previous section. Besides the RMS error, L_1 and L_∞ errors are calculated and show a nearly identical pattern, differing only in the vertical scale. The obvious difference is that 16-bit POSITs have about 1/4 the rounding error of floats when running algorithms that round at identical places in the data flow. This follows from the fact that most of the operations occur where POSITs have two bits greater accuracy in the fraction part of their format.

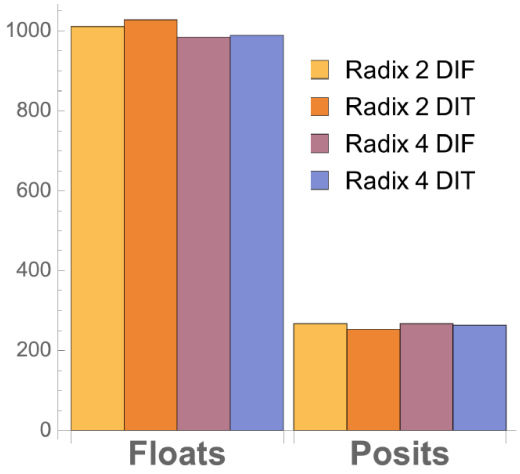


Figure 2: RMS errors per value $\times 10^6$ for 1024-point FFTs.

POSITs, on the other hand, 97.9 percent of the values make the round trip with all bits identical to the original value. Of the 2.1 percent that are off, they are off by only 1 ULP. While the reversibility is not mathematically perfect, it is nearly so, and may be accurate enough to eliminate the need to use 32-bit data representation.

Company Description

NSCC Singapore was established in 2015 and manages Singapore's first national petascale facility with available high performance computing (HPC) resources. As a National Research Infrastructure funded by the National Research Foundation (NRF), they support the HPC research needs of the public and private sectors, including research institutes, institutes of higher learning (IHLs), government agencies and companies.