IMPROVING FIXED-POINT ACCURACY OF FFT CORES IN O-OFDM SYSTEMS

Robert Koutsoyannis[1], Peter A. Milder[1], Christian R. Berger[1], Madeleine Glick[2], James C. Hoe[1], and Markus Püschel[3]

[1] Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, USA
[2] APIC Corporation, Culver City, CA, USA
[3] Department of Computer Science, ETH Zurich, Switzerland

ABSTRACT

Optical OFDM communication systems operating at data rates in the 40Gb/s (and higher) range require high-throughput/highly parallel fast Fourier transform (FFT) implementations. These consume a significant amount of chip resources; we aim to reduce costs by improving the system's accuracy per chip-area. For OFDM signals, we characterize the growth of data within the FFT and explore several cost-conscious methods for improving the fixed-point format. Using ASIC synthesis and hardware accurate simulations, we evaluate the corresponding system error and stability of these methods. We introduce *Directive Scaling*, which provides an average increase in overall accuracy without additional runtime-adaptive mechanisms. ASIC synthesis results show minimal overhead, and we explicitly evaluate and explain the inherent tradeoffs. When applied to an 8-bit IFFT design, our technique improves precision by approximately two bits with just a 4% area overhead, as opposed to the additional 32% area overhead required using standard methods.

Index Terms—*OFDM, Fixed-Point, FFT, Saturation*

1. INTRODUCTION

Orthogonal frequency-division-multiplexing (OFDM) has become ubiquitous in communications across frequency selective channels. It boasts high spectral efficiency and resilience to channel impairments. OFDM now dominates wireless communications, e.g. WiFi (IEEE 802.11a/g), and is now considered for optical fiber communication systems [5]. Unlike wireless however, optical fiber systems have signal bandwidths in the 10-100GHz spectrum and push for data rates higher than 40Gb/s. Such high throughput demands are only achievable with highly parallel hardware. Therefore, the cost of the DSP, particularly the fast Fourier transform (FFT), rapidly becomes a first order issue.

For example, the FPGA-based OFDM transmitter developed in [3] requires a throughput of one 128-point IFFT every clock cycle at 167MHz to generate an 8.34 Gb/s QPSK-OFDM signal. The 10-bit fixed-point FFT implementation consists of 2,308 parallel adders and 908 parallel multipliers—consuming over 75% of the Virtex-4 FX100 FPGA's resources. Current systems [4] target even higher data rates with larger FFT sizes, faster clocks, and even more data precision to handle higher QAM modulation formats—each of which compound the cost.

By improving the efficiency of the fixed-point format within the FFT, one can reduce the number of bits required, resulting in decreased area and power consumption. For example, our 12-bit FFT design (synthesized for 65nm ASIC) requires 26% more area than a 10-bit design. The 12-bit FFT FPGA implementation in [3] requires 20% more area than its 10-bit counterpart. Although the rate of savings diminishes with higher bit-precision, the target accuracy for O-OFDM systems is within this range. This is because any added precision would be dominated by the error introduced by the communications channel and limited resolution DACs and ADCs.

As data is computed in the discrete stages of FFT, the largest value increase (in magnitude) that can occur is a factor of two per stage, while the average magnitude increases only by a factor of $\sqrt{2}$ [2]. Conventional FFT implementations therefore employ *forced scaling*, which scales the data by a factor of ½ (one bit-shift right) after each stage to fully avoid overflow. However, relative to the average magnitude, this reduces the fixed-point precision by one half of a bit per stage.

Alternatively, at additional cost, adaptive hardware mechanisms like block floating point can be used to *conditionally scale* the data, i.e. scale data only when an overflow is detected [2]. So, when overflow does not occur in a stage, scaling will not be performed, and the least significant bit for each value is preserved. This adaptive method avoids overflow at all cost, even if just one or very few data elements overflow.

For OFDM signal inputs to the FFT, there is a low probability of overflow in nearly half the stages, while there is high probability in the others. In this paper, we introduce *directive scaling*, a technique that takes advantage of this predictable pattern. Directive scaling works by using forced scaling in FFT stages likely to overflow; in the other stages we tolerate an occasional overflow by using saturating arithmetic. We perform ASIC synthesis to quantify hardware costs and use hardware-accurate MATLAB simulations to quantify the numerical error of directive scaling and existing techniques. We show that our strategy approaches the accuracy of conditional scaling but with implementation cost very close to the inexpensive forced scaling method.

2. O-OFDM AND THE FFT HARDWARE

The fast Fourier transform (FFT) is the most expensive DSP component in an Optical Orthogonal Frequency Division Multiplexing (O-OFDM) transceiver. A simplified OFDM transceiver is shown in Figure 1(a). The transmitter generates a complex baseband signal by modulating data symbols, e.g., quadrature phase-shift keying (QPSK) or quadrature amplitude modulation (QAM) (see Figure 1(b)) onto frequency subcarriers using an inverse discrete Fourier transform (IDFT) of size n,

$$x[k] = \sum_{l=0}^{n-1} s[l] e^{j2\pi\frac{lk}{n}}. \tag{1}$$

Several aspects of the fast Fourier Transform (FFT) hardware implementation contribute to its fixed-point accuracy. The following sections provide insight and arguments for several design choices.

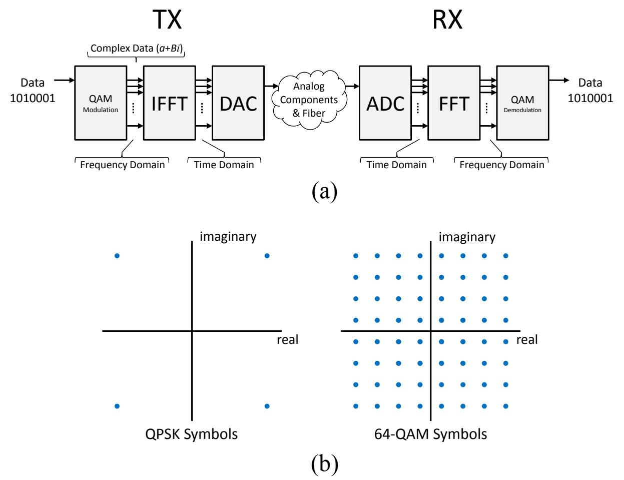


Figure 1. (a) Simplified OFDM Transceiver, (b) Complex Data Constellations

Pease FFT Algorithm. Many FFT algorithms exist for efficiently computing the discrete Fourier transform (DFT) and its inverse (IDFT). In this paper we consider the Pease FFT [7], which is frequently used in hardware implementations of the DFT due to its regular structure. This algorithm can be realized as several different types of datapaths, each with a different cost/performance tradeoff [8].

Figure 2 shows an example of the dataflow of the Pease FFT with n = 16 points and radix r = 4. The radix 4 Pease FFT on $n = r^t$ points consists of t stages of parallel DFT_r instances (each of which perform an FFT on r points), data reordering stages (permutations), and scaling by complex phasors (twiddle factors). Each of the DFT_4 blocks is further decomposed into DFT_2 blocks as shown. The DFT_2 blocks, often called butterflies, contain only one addition and one subtraction.

Each non-trivial ($\neq 1$) twiddle multiplication can contribute significant error. The irrational root-of-unity twiddle values are stored as p-bit values with added quantization error. As the radix of an FFT algorithm increases, the number of nontrivial twiddle factors decreases, but with quickly diminishing returns. For example, a 256 point radix-4 FFT has 22.3% fewer significant twiddles than a radix-2 FFT of the same size. Increasing the radix further to 16 yields only an additional 4.4% reduction. Note however, as the radix r of an algorithm increases, fewer problem sizes r^t can be directly represented with this manner of algorithm. Therefore, this paper considers the radix 4 Pease FFT algorithm, but it can be extended to a wider space of radices and algorithms.

Fixed-Point Representation. The fixed-point representation and associated arithmetic of the data within the FFT implementation dictate the numerical accuracy. Each complex number is represented by two data words, one for each of the real and imaginary portions of the number. Each word is a p-bit 2's complement value normalized to the bounds [–1,1) or more accurately, $[-1,\ 1-(½)^{(p-1)}]$. In this fractional 2's complement format, the most significant bit has a value of –1, followed by 0.5, 0.25,... and so on to the least significant bit $(½)^{(p-1)}$. We assume that all operations that result in a rounding of data are handled with simple truncation, i.e. rounding towards negative infinity.

Scaling Options within the FFT. Regardless of the choice of radix r, each data sample must pass through $\log_2 n$ DFT_2 butterflies (for example, 4 stages in Fig 2). Each time a pair of data elements

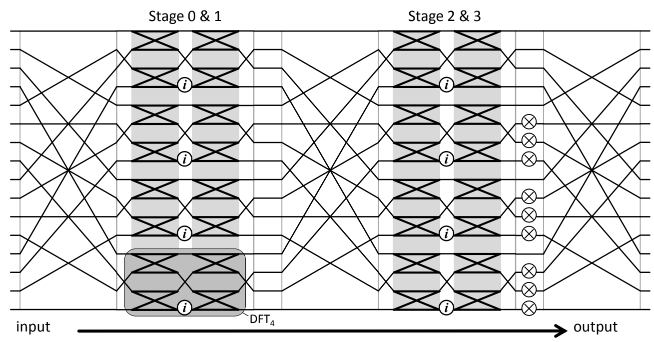


Figure 2. Radix 4 Pease FFT for n = 16

passes through one of these stages (also potentially including a twiddle factor), its values grow in magnitude [2]. We write the computation of one stage as

$$y[2i] = (\,x[2i] + x[2i+1]\,)\,\omega_{2i}, \qquad y[2i+1] = (\,x[2i] - x[2i+1]\,)\,\omega_{2i+1} \quad (2)$$

where y is the output of the stage, x is the input, and the $|\omega| = 1$ represent complex twiddle factors. Here, $y[i]$ can never be larger than twice the magnitude of $x[i]$. More specifically,

$$\max_i\{|x[i]|\} \leq \max_i\{|y[i]|\} \leq 2 \cdot \max_i\{|x[i]|\}. \quad (3)$$

Therefore, to avoid overflow within a stage of the FFT, the most common fixed-point hardware designs either increase the bit precision p by one bit after each butterfly, e.g. [6], or shift all data words right by one bit (scale by ½) as in [2].

On the other hand, from the variance of the values,

$$E[|y[i]|^2] = 2 \cdot E[|x[i]|^2], \quad (4)$$

where we assume that the $x[i]$ are zero-mean and independent and identically distributed (i.i.d.) OFDM symbols. This means that the standard deviation, or the average magnitude of the values, will grow with a factor of only $\sqrt{2}$ per stage. Thus while the maximum possible magnitude grows with a factor of two per stage, the average only grows "half as fast" in a logarithmic sense.

Considering this, most values will not need scaling at every stage to avoid overflow—in fact, it's often the case that none do. Thus, by forcing scaling, i.e. avoiding overflow at all cost, precision is unnecessarily lost.

Existing adaptive scaling mechanisms (such as block floating point) greatly improve accuracy by determining at run-time whether or not scaling is necessary for each stage, ultimately avoiding unnecessary truncation [2]. Several variations of adaptive mechanisms can be realized based on the conditions for scaling and they each have unique accuracy/cost tradeoffs.

In this paper, we will consider the two most common methods of scaling data to avoid overflow (one fixed, and one adaptive):

1. *Forced Scaling (FS):* Data words are shifted right or equivalently scaled by a factor of ½ after each butterfly.
2. *Conditional Scaling (CS):* After each stage of butterflies all n values are compared to a threshold; if one or more values exceed this bound, *all* values are scaled by a factor of ½.

Scaling decisions are only made at the output of the butterflies. Therefore, we must be careful in avoiding overflow at the complex multipliers due to rotations. This is easily prevented by limiting the input data size and CS threshold to $\sqrt{2}/2$ or less—keeping the values within 'region A' of Figure 3. Figure 3 shows

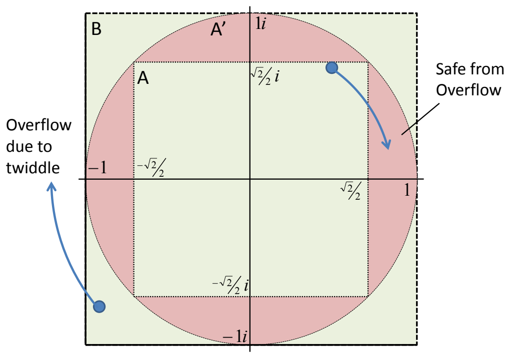


Figure 3. Fixed-Point and Overflow Regions, similar to [2]

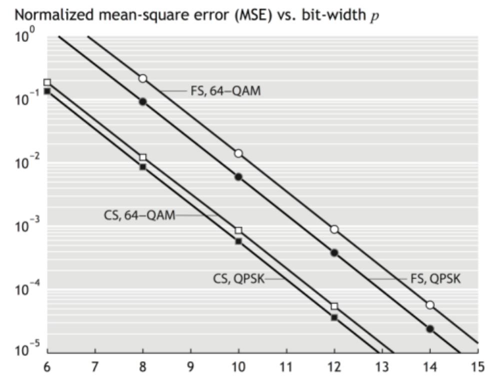
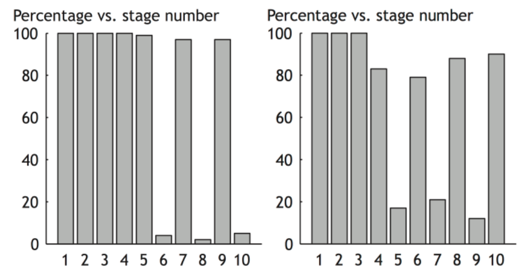


Figure 4. NMSE of Forced and Conditional Scaling IFFT_{1024}



(a) n = 1024, QPSK (b) n = 1024, 64-QAM

Figure 5. Conditional Scaling Decisions per Stage

how a complex twiddle multiplication can rotate values out of the fixed-point bounds, causing overflow. Alternatively, this paper also reports results with a threshold of 0.5 because in hardware, comparing every individual data word to a power of 2 constant requires less logic than a p-bit constant representing $\sqrt{2}/2$. We demonstrate this later where we explicitly compare the hardware cost of the two boundary conditions.

3. FFT FIXED-POINT ACCURACY IN OFDM SYSTEMS

The numerical error introduced by the fixed-point implementation of FFTs depends significantly on the considered input data. In this paper we are particularly interested in the error introduced by an IFFT as part of an optical orthogonal frequency-division-multiplexing (O-OFDM) communications system; here the inputs are i.i.d. randomly drawn QAM symbols, (see Figure 1(b)).

As a measurement setup, we compare our p-bit fixed-point IFFT designs with an ideal (double-precision floating point) IFFT. The mean-square error (MSE) of the signal is then, following from eqn. (1),

$$MSE = E[|\tilde{x}[k] - x[k]|^2], \tag{5}$$

which we assume independent of k. The normalized MSE corresponds to the inverse signal-to-noise ratio at the transmitter.

We created a hardware-accurate MATLAB simulation to evaluate the normalized MSE caused by a p-bit fixed-point IFFT, and show its results in Figure 4. In general all error curves decrease by about one decade per two bits of resolution. We show errors to 10^{-5}, because below this the error is dominated by the signal converters and the communications channel.

From Figure 4, by comparing the curves at an identical NMSE value, we can see that up to two bits of precision can be gained by using CS. Since CS fully avoids overflows, its performance will strictly be better than that of the conventional FS implementation (we are only making better use of the available fixed-point representation). Our goal is to then capitalize on the obvious room for improvement, but without resorting to the additional complexity of run-time methods. The additional overhead of CS incurs a non-trivial hardware cost which we present later. Accordingly, any adaptive method that determines the optimum bit representation of data at run-time would incur a similar cost.

4. DIRECTIVE SCALING

Figure 5 illustrates the run-time decisions made by CS while computing 100 IFFTs with (a) QPSK and (b) 64-QAM input signals. Each bar shows the percentage of time the IFFT needed to scale in each of the 10 stages. For example, we see that with QPSK inputs, the CS unit always needed to scale in stage 1, but rarely in stage 10. Note that the result for each stage is dependent on all previous stages. After several stages of scaling, the values are reduced so they rarely overflow in a particular stage (e.g. Figure 5(a), stage 6). Then, in subsequent stages, due to the average growth per stage of $\sqrt{2}$, the data begin a pattern where they must be scaled in every other stage only. Notice how the pattern begins at an earlier stage for 64-QAM. This is because the average magnitude of the input signal is lower for 64-QAM than QPSK.

As shown in [1], OFDM signals begin to look Gaussian as they progress through the IFFT. At that point, we characterize the signals in terms of their standard deviation which correlates with the maximum fixed-point representation. Furthermore, since we know the IFFT grows by $\sqrt{2}$ per stage, scaling essentially becomes predictable with a probabilistic confidence.

It should also be pointed out that when determining a priori whether a particular stage should scale, the following stages will be affected by a "ripple effect" compared to the CS behavior. For example, in Figure 5(b), if we fix stage 4 to always scale, we antedate one scaling decision for about 20% of trials, which will lead them to skipping their next scaling decision instead. Hence stage 5 would no longer need to scale and every data element would preserve the least significant bit.

From these observations we introduce *Directive Scaling* (DS), where we a priori determine for each stage whether or not to scale. In order to tolerate the rare occasions where a value grows too large in a non-scaling stage, we use saturating adders to clip values to their maximum—incurring instead a small saturation error as opposed to a 2's complement overflow error. For a given system we choose a *directive strategy*, a length $\log_2 n$ vector of scaling

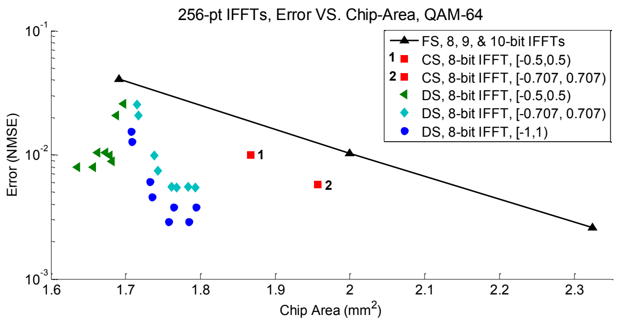


Figure 6. IFFT designs comparing scaling techniques

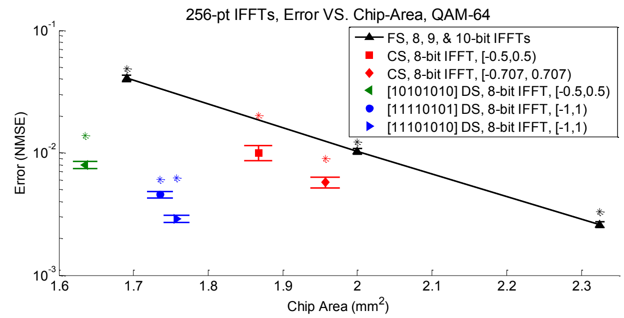


Figure 7. IFFT designs showing stability of error

decisions, where 1 indicates a scaling stage and 0 indicates a saturating stage, e.g. [110101...]. The choice of scaling pattern depends on several factors, including the bit precision, number of (non-zero) IFFT inputs, input signal average magnitude, IFFT size, and QAM modulation format. Given these system parameters, the directive is easily found from running simulations at design time, producing the scaling graphs like Figure 5.

Saturating logic must be added to mitigate the rare overflows in DS, adding cost to the baseline FS hardware design. Nevertheless, the cost is significantly less than the adaptive mechanisms used in CS. Like CS, saturating logic allows choices for a threshold value. DS can use a threshold of ~0.707 to prevent overflow in the complex multipliers, or it can use the full fixed-point boundary, [–1, 1) if saturating logic is included in the multipliers in addition to the adders.

5. EVALUATION

In this section we evaluate the numerical error and chip area cost of IFFT designs using directive scaling, and compare the results with forced and conditional scaling. We find chip area by creating RTL Verilog implementations of each design (based on FFTs generated with Spiral [8]) and synthesizing them using Synopsys Design Compiler targeting a 65nm standard cell library. Each design successfully met all timing constraints at 200MHz, which provides sufficient throughput for real-time transceivers (e.g., [3, 4]). Then for each design we use the previously mentioned bit-accurate simulation to find the normalized mean square error (NMSE) using 1000 trials with QAM-64 input signals.

Figure 6 shows the ASIC area (x-axis) vs. numerical error (y-axis) of several scaling strategies for 256-point, 8-bit IFFTs. First, the black line with triangles shows our baseline: forced scaling with 8, 9, and 10 bits of precision (from left to right). As expected, we see that increasing precision reduces error while increasing area. Next, we show two red squares for conditional scaling (with two different thresholds). We observe that these points improve upon forced scaling; the 8-bit CS designs provide lower error than forced scaling with lower area.

Next, the diamonds, triangles, and squares show three different DS designs. Each group uses different thresholds (.5, .707, or 1, as explained in Section 4). Within a group, the differences between the points correspond to different choices of directive (e.g. [11101010] vs. [11110101]). We see that the DS designs improve upon both forced and conditional scaling; they can provide higher accuracy at lower cost.

Figure 7 shows a more in-depth error analysis for a subset of designs. Each point now has error bars representing 1 standard deviation in error and a star showing the worst observed error in 1,000 trials. Here we show how DS designs are on par with the error of CS, yet have more relative stable error. This is because although CS adaptively performs better than FS, occasionally a combination of inputs forces additional error. The stable error seen for the FS designs makes sense considering there's no variability when scaling at each stage.

Although we show DS operating on the IFFT, we have also successfully applied the technique to the FFT (receiver). Here the directives are reversed, where the initial stages alternate scaling and saturation, while the later stages require scaling throughout.

6. CONCLUSION

When generating OFDM signals, the growth of data in the IFFT follows a predictable pattern. This paper introduced *directive scaling*, which exploits this predictability to produce an IFFT design that scales data in stages where overflow is likely and tolerates occasional overflow elsewhere. This technique improves on forced and conditional scaling, matching or exceeding their accuracy at lower cost.

7. REFERENCES

[1] J. Armstrong, H. A. Suraweera, S. Brewer, R. Slaviero, "Effect of Rounding and Saturation in Fixed-Point DSP Implementation of IFFT and FFT for OFDM Applications," *The Embedded Signal Processing Conference* (GSPx 2004), Sept. 2004.

[2] P. D. Welch, "A fixed-point fast Fourier transform error analysis," *IEEE Trans. Audio and Electroacoustics*, vol. AU-17, no. 2, pp. 151-157, June 1969.

[3] Y. Benlachtar, P. Watts, R. Bouziane, P. Milder, R. Koutsoyannis, J. Hoe, M. Püschel, M. Glick, and R. Killey, "Real-time digital signal processing for the generation of optical orthogonal frequency-division-multiplexed signals," *IEEE J. Select. Topics Quantum Electronics*, vol. 16, no. 5, pp. 1235-1244, Sept. 2010.

[4] R. Schmogrow, M. Winter, D. Hillerkuss, B. Nebendahl, S. Ben-Ezra, J. Meyer, M. Dreschmann, M. Huebner, J. Becker, C. Koos, W. Freude, and J. Leuthold, "Real-time OFDM transmitter beyond 100 Gbit/s," Opt. Express, vol. 19, no. 13, pp. 12740-12749, Jun. 2011

[5] J. Armstrong, "OFDM for optical communcations," *J. Lightwave Technol.*, vol. 27, no. 3, pp. 189–204, Feb. 2009.

[6] W.-H. Chang and T. Q. Nguyen, "On the fixed-point accuracy analysis of FFT algorithms," *IEEE Trans. Signal Processing*, vol. 56, no. 10, pp. 4673–4682, Oct. 2008.

[7] M. C. Pease, "An adaptation of the fast Fourier transform for parallel processing," *J. of the ACM*, vol. 15, no. 2, April 1968.

[8] P. A. Milder, F. Franchetti, J. C. Hoe, and M. Püschel, "Formal datapath representation and manipulation for implementing DSP transforms," in *Proc. Design Automation Conference (DAC)*, 2008, pp. 385–390.