On the Fixed-Point Accuracy Analysis of FFT Algorithms

2 Author(s)

Wei-Hsin Chang ; Truong Q. Nguyen

View All Authors

Abstract:
In this paper, we investigate the effect of fixed-point arithmetics with limited precision for different fast Fourier transform (FFT) algorithms. A matrix representation of error propagation model is proposed to analyze the rounding effect. An analytic expression of overall quantization loss due to the arithmetic quantization errors is derived to compare the performance with decimation-in-time (DIT) and decimation-in-frequency (DIF) configurations. From the simulation results, the radix-2 DIT FFT algorithm has better accuracy in term of signal-to-quantization-noise ratio (SQNR). Based on the results, a simple criterion of wordlength optimization is proposed to yield comparable accuracy with fewer bit budget.

Published in: IEEE Transactions on Signal Processing (Volume: 56, Issue: 10, Oct. 2008)

Page(s): 4673 - 4682

INSPEC Accession Number: 10222474

Date of Publication: 16 September 2008

DOI: 10.1109/TSP.2008.924637

ISSN Information:

Publisher: IEEE

Sponsored by: IEEE Signal Processing Society

SECTION I.
Introduction

The discrete Fourier transform is one of the fundamental operations in digital signal processing. It is widely used in noise reduction, global motion estimation [1], and orthogonal-frequency-division-multiplexing (OFDM) systems such as wireless LAN (802.11a) [2], digital video broadcasting (DVB), digital audio broadcasting (DAB), and MB-OFDM [3]–[4][5]. The original computation of discrete Fourier transform (DFT) with N-sample input requires N^2 complex multiplications. Cooley and Tukey [6] first introduced the concept of fast Fourier transform (FFT) to demonstrate a significant computational reduction from $O(N^2)$ to $O(N \log N)$ by making efficient use of symmetry and periodicity properties of the twiddle factors. Since the pioneering work of Cooley–Tukey algorithm [6], several algorithms have been developed to further reduce the computational complexity, including radix-4 [7], radix-2^2 [8] and split-radix [9]. In general, these fast algorithms divide the FFT computation into odd- and even-half parts recursively and then extract as many common twiddle factors as possible. The number of required real additions and multiplications is usually used to compare the efficiency of different FFT algorithms. A modified split-radix algorithm proposed by Johnson and Frigo [10] has the fewest number of operation counts.

In general, considering the actual hardware design, the accuracy of FFT/IFFT module is an important design factor of system performance. In practice, fixed-point arithmetic is used to implement FFT algorithms in hardware because it is not possible to keep infinite resolution of coefficients and operations. All coefficients and input signals have to be represented with finite number of bits in binary format depending on the tradeoff between the hardware cost (memory usage) and the accuracy of output signals. Generally speaking, each multiplication may introduce an error due to rounding operations or truncations, which is referred as arithmetic quantization error. Besides, all the twiddle factors are represented with limited number of bits and the loss due to the inexact coefficients is called coefficient quantization error. The theoretical performance evaluation has been given in previous works. Several previous works have analyzed the effect of fixed-point arithmetic for radix-2 FFT [11]–[12][13][14][15]. James [16] derived the fixed-point MSE analysis of quantization loss for mixed-radix FFT algorithms with conventional complex multipliers. Perlow and Denk [17] proposed an error propagation model to estimate the performance for radix-2 FFT architecture. Instead of conventional implementation of complex multiplier, Chandra [18] studied the same issue for radix-2 DIT FFT in the logarithmic numbering system. Park and Cho [19] used a propagation model to address the error analysis for CORDIC implementations. Chang [20] also gave an experimental comparison of FFT computation with different implementations of complex multiplier. However, since split-radix FFT and other higher-radix FFT algorithms are proved to be computationally efficient than radix-2 [9] and are widely used for hardware implementation [21], [22], as such the study of finite precision effect of all FFT algorithms is of increasing significance.

In this paper, the arithmetic quantization error is analyzed and quantitatively compared for different FFT algorithms. The rest of this paper is organized as follows. In Section II, several FFT algorithms including radix-2, radix-4 and split-radix algorithms are reviewed and compared in terms of their computational efficiency. The previous works on the fixed-point effect of FFT quantization loss are also cited. In Section III, a propagation model in matrix form of both DIF and DIT FFT algorithms is proposed to calculate the overall noise power resulting from the arithmetic quantization errors. The analytic results of different FFT decompositions are presented. An optimization scheme of memory wordlength adjustment is also proposed for short-length FFT architecture. In Section IV, the fixed-point simulation of all FFT algorithms is illustrated. Finally we conclude this work in Section V.

SECTION II. Previous Work

A. FFT Algorithms

In this section, the radix-2, radix-4 and split-radix FFTs are reviewed. Given an input sequence $x(n)$, the N-point DFT is defined as

$$X(k) = \sum_{n=0}^{N-1} x(n) \cdot W_N^{nk}, \quad k = 0, 1, \ldots, N-1. \tag{1}$$

View Source

where n is the time index, k is the frequency index, and the twiddle factor W_N^{nk} is defined as

$$\begin{aligned} W_N^{nk} &= \exp\left(\frac{-2j\pi nk}{N}\right) \\ &= \cos\left(\frac{2\pi nk}{N}\right) - j \cdot \sin\left(\frac{2\pi nk}{N}\right). \end{aligned} \tag{2}$$

View Source

A larger FFT can be decomposed into cascaded FFT stages with smaller length. Assume the N-point FFT is partitioned into α stages and define

$$N_\beta = \frac{N}{r_1 \cdot r_2 \cdots r_\beta} \quad \text{where} \quad 1 \le \beta \le \alpha - 1 \tag{3}$$

View Source

the final stage is derived as [23]

$$\begin{aligned} &X(r_1 r_2 \cdots r_{\alpha-1} m_\alpha + r_1 r_2 \cdots r_{\alpha-2} m_{\alpha-1} \\ &\quad + \cdots + r_1 m_2 + m_1) \\ &\quad = \sum_{} \end{aligned}$$

$$= \sum_{q_{\alpha-1}=0} x_{\alpha-1}(q_{\alpha-1}, m_{\alpha-1}) W_{r_\alpha} \quad . \tag{4}$$

View Source

The intermediate βth stage, $x_\beta(q_\beta, m_\beta)$ is given by the recursive equation as

$$x_\beta(q_\beta, m_\beta) = W_{N_{\beta-1}}^{q_\beta m_\beta} \sum_{p=0}^{r_\beta - 1} x_{\beta-1}(N_\beta p + q_\beta, m_{\beta-1}) W_{r_\beta}^{p m_\beta} \tag{5}$$

View Source

where $2 \le \beta \le \alpha - 1$, $0 \le m_i \le r_i - 1$, $0 \le q_i \le N_i - 1$, and $2 \le i \le \alpha$.

Each summation represents a r_β-point FFT computation. Please note the above decomposition procedure is not unique. There are various equivalent decompositions for the same N-point FFT. As a result, many designs of memory-based FFT architecture will decompose a larger FFT computation into several cascaded smaller FFTs and utilize a single FFT core to reduce the hardware cost [24]. There are two alternative types of FFT algorithms, including DIT FFT and DIF FFT. The former decimates the input samples in time domain to generate sequential transformed output in frequency domain, and the latter acts in a reverse behavior.

The idea of Cooley–Tukey radix-2 algorithm [6] is to divide the original operation into two half-length FFTs and take advantage of extracting common terms to reduce the number of multiplications. When the length of the input sequence $x(n)$ is power of 4, the radix-4 FFT algorithm [7] is derived by extracting more trivial complex multiplications and applying the periodicity property ($W_N^{(kN/4)} = (-j)^k$, $W_N^{(kN/2)} = (-1)^k$, and $W_N^{(3kN/4)} = (j)^k$). Note that the input to each (N/4)-point DFT is a linear combination of four samples scaled by twiddle factors. This procedure is repeated $\log_4 N$ times. In [9], as with the radix-2 FFT case, the DFT operation is first split into odd-half and even-half parts. The odd components are further decomposed into $4k + 1$ and $4k + 3$ frequency components. Repeating this process for the half- and quarter-length DFTs gives the split-radix FFT algorithm. The comparison between these algorithms is made in terms of the number of real multiplications as shown in Table I. For FFT operations whose length is not power of 4, a mixed-radix scheme consist of radix-4 stages and an additional radix-2 stage is used.

Table I Number of Nontrivial Twiddle Factors

FFT Size	Radix-2	Radix-4	Split-Radix
8	2	2	2
16	10	8	8
32	34	28	26
64	98	76	72
128	258	204	186
256	642	492	456
512	1538	1196	1082
1024	3586	2732	2504
2048	8194	6316	5690
4096	18434	13996	12744

B. Effect of Finite Precision in Digital Systems

The effect of finite wordlength in digital systems has been studied during the past decades. Weinstein [25] classified the quantization loss into four categories including quantization of coefficients, error due to A/D conversion, roundoff noise due to arithmetic rounding operations, and the constraint on signal level to maintain the dynamic range and prevent overflow. Oppenheim [11] analyzed the effect of finite precision for digital filtering and radix-2 DIT FFT but did not consider the fact that some twiddle factors are trivial and do not contribute any noise. Thong and Liu [12] also investigated the same phenomenon for both DIT and DIF radix-2 FFT with neglect of the quantization of coefficient because it is shown the error due to the quantization of coefficient is less significant than that of arithmetic rounding operations [25].

The additive noise model of quantization loss is widely adopted to measure the effect of the fixed

The additive noise model of quantization loss is widely adopted to measure the effect of the fixed length operations in digital signal processing systems [7], [11]. The quantized product can be expressed as the sum of unquantized product and an uniformly-distributed additive quantization noise. Consider the multiplication of quantized numbers $\hat{x}$ and $\hat{a}$, the product y itself will be quantized to a $(B+1)$-bit number, $\hat{y} = Q_B[y]$. The quantized product term can be expressed as the unquantized product with an additive quantization noise source, e, as depicted in Fig. 1. For rounding errors represented in two's complement format, assuming that e is a uniformly distributed random variable whose pdf is shown in (6), its variance can be easily calculated as $\sigma^2 = (\Delta^2)/(12)$, where $\Delta = 2^{-B}$

$$p(e) = \begin{cases} \frac{1}{\Delta}, & -\frac{\Delta}{2} < e \leq \frac{\Delta}{2} \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

View Source

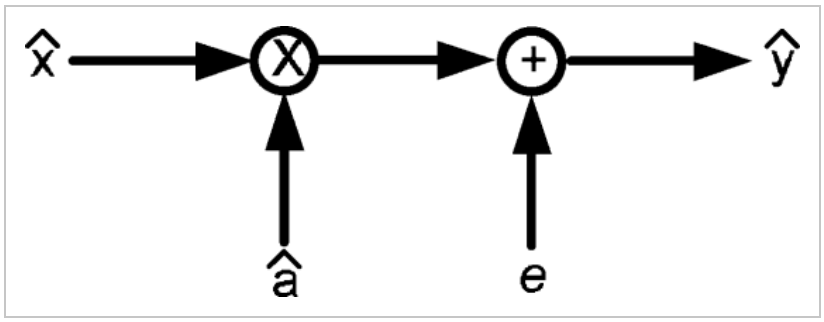

Fig. 1.
Additive noise model of quantization loss.

For a conventional complex multiplier, assume that $x = x_r + jx_i$ represents the input sample, and that $t = c + js$ represents the complex twiddle factor. Then the quantized output is represented as follows:

$$Q(Y) = [Q(cx_r) - Q(sx_i)] + j[Q(cx_i) + Q(sx_r)]. \tag{7}$$

View Source

Consequently, the overall quantization loss and the variance of quantization loss are expressed as

$$Q(Y) - Y = (e_{y1} - e_{y2}) + j(e_{y3} + e_{y4}) \tag{8}$$

$$\begin{aligned} E_c[|Q(Y) - Y|^2] &= E[(e_{y1} - e_{y2})^2 + (e_{y3} + e_{y4})^2] \\ &= E[e_{y1}^2] + E[e_{y2}^2] + E[e_{y3}^2] + E[e_{y4}^2] \\ &= 4\sigma^2 \end{aligned} \tag{9}$$

View Source

where $e_{y1} = Q(cx_r) - cx_r$, $e_{y2} = Q(sx_i) - sx_i$, $e_{y3} = Q(cx_i) - cx_i$, and $e_{y4} = Q(sx_r) - sx_r$ are all real-valued random variables defined in (6). The variance of other implementations of complex multipliers can be found in [20].

SECTION III.
Matrix Representation of FFT Algorithms

In this section, we derive the equivalent matrix form of both DIF and DIT FFT algorithms. Although the alternative DIT and DIF FFT algorithms have the same multiplicative complexity, the sequence of butterfly stages and twiddle factor stages is reversed. In other words, the signal flow of two alternative representations is actually the mutual mirroring of each other.

A. Propagation Model of DIF FFT Algorithms

For an N-point DIF FFT computation as shown in Fig. 2, define $\alpha = \log_2 N$, we can rewrite DIF FFT in the matrix representation as follows:

$$X_F = \prod_{i=0}^{\alpha-1} w_{F_i} B_{\alpha,\alpha-i} x \tag{10}$$

View Source

where each matrix is explained as follows:

1. x is the $N \times 1$ input vector;
2. X_F is the $N \times 1$ transformed output vector;
3. $B_{\alpha,\alpha-i}$ is the $N \times N$ equivalent butterfly matrix at the ith stage of 2^α-point DIF FFT.
4. w_{F_i} is the $N \times N$ equivalent twiddle factor matrix at the ith stage, where w_{F_i} is a diagonal matrix whose elements are the twiddle factors of the ith stage.

Take an 8-point DIF FFT as example, the corresponding $B_{\alpha,\alpha-i}$ matrices are defined below. The w_{F_i} matrices of the radix-2 DIF FFT algorithms are also listed. Please note the variation of w_{F_i} as the radix of FFT algorithms changes.

$$B_{3,3} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \end{bmatrix} = I_1 \otimes B_{3,3} \tag{11}$$

$$B_{3,2} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 \end{bmatrix} = I_2 \otimes B_{2,2} \tag{12}$$

$$B_{3,1} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix} = I_4 \otimes B_{1,1} \tag{13}$$

$$B_{1,1} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = H_2 \tag{14}$$

$$w_{F_0} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & W_8^1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & W_8^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & W_8^3 \end{bmatrix} \tag{15}$$

$$w_{F_1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & W_8^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{16}$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & W_8^2 \end{bmatrix}$$

View Source

where H_2 is the 2×2 Walsh matrix, $\otimes$ denotes Kronecker product, and I_p is the p $\times$ p identity matrix. From (11)–(14), the $B_{\alpha,i}$ matrices can be generalized as

$$B_{i,i} = H_2 \otimes I_{2^{i-1}} \tag{17}$$

$$B_{\alpha,i} = I_{2^{\alpha-i}} \otimes B_{i,i}, \quad \alpha \geq i. \tag{18}$$

View Source

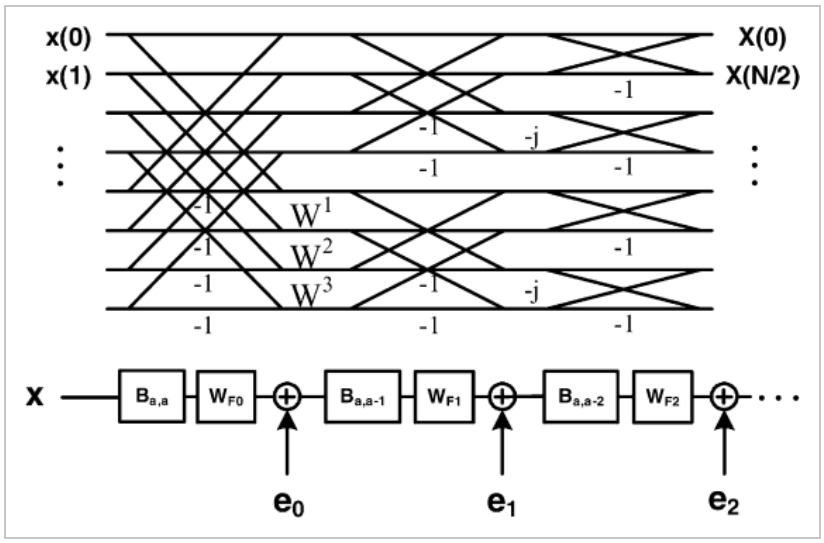

Fig. 2.
Propagation model of quantization loss for DIF FFT.

Define $e_i(0 \leq i \leq \alpha)$ as the corresponding $N \times 1$ additive noise vector of w_{F_i}. Its elements, $e_i(j)(0 \leq j \leq N-1)$, are uncorrelated complex uniform random variables with zero mean and variance equal to σ_c^2, if a nontrivial complex multiplication is presented at $w_{F_{i,j}}$. An example for the second stage of 8-point DIF split-radix FFT is as follows:

$$e_1 = [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad e_1(5) \quad 0 \quad e_1(7)]_t. \tag{19}$$

View Source

Let x_i be the intermediate result after i stage, $\tilde{x_i}$ be the corresponding erroneous version with quantization loss due to nontrivial complex multiplications, and Δx_i be the difference between the original output and the erroneous output. Therefore, all x_i and Δx_i can be expressed as

$$\tilde{x_0} = w_{F_0} B_{3,3} x + e_0 \tag{20}$$

$$\begin{aligned} \tilde{x_1} &= w_{F_1} B_{3,2} \tilde{x_0} + e_1 \\ &= w_{F_1} B_{3,2} (w_{F_0} B_{3,3} x + e_0) + e_1 \end{aligned} \tag{21}$$

$$\begin{aligned} \tilde{x_2} &= w_{F_2} B_{3,1} \tilde{x_1} + e_2 \\ &= w_{F_2} B_{3,1} [w_{F_1} B_{3,2} (w_{F_0} B_{3,3} x + e_0) + e_1] + e_2 \end{aligned} \tag{22}$$

$$\Delta x_0 = e_0 \tag{23}$$

$$\Delta x_1 = w_{F_1} B_{3,2} e_0 + e_1 \tag{24}$$

$$\Delta x_2 = w_{F_2} B_{3,1} w_{F_1} B_{3,2} e_0 + w_{F_2} B_{3,1} e_1 + e_2. \tag{25}$$

View Source

Based on the observation of the recursive representation shown from (23)–(25), we can find the general expression of the overall quantization loss, ΔX_F, as

$$\Delta X_F = \Delta x_{\alpha-1} = \sum_{i=0}^{\alpha-1} \left(\prod_{j=i+1}^{\alpha-1} w_{F_j} B_{\alpha,\alpha-j} \right) e_i \tag{26}$$

View Source

The subjective is to calculate the total noise power of the quantization loss. We present the

following lemmas which will be useful in later derivation.

Lemma 1

The matrix product of $B_{\alpha,i}$ and its own Hermitian matrix equals to $2I_{2^\alpha}$.

Proof

$$\begin{aligned} B_{\alpha,i}B_{\alpha,i}^H &= (I_{2^{\alpha-i}} \otimes B_{i,i})(I_{2^{\alpha-i}} \otimes B_{i,i})^H \\ &= (I_{2^{\alpha-i}} \otimes B_{i,i}) \left(I_{2^{\alpha-i}}^H \otimes B_{i,i}^H\right) \\ &= I_{2^{\alpha-i}} \otimes (B_{i,i}B_{i,i}^H) \\ &= I_{2^{\alpha-i}} \otimes \left(H_2 \otimes I_{2^{i-1}}\right) \left(H_2^H \otimes I_{2^{i-1}}^H\right) \\ &= I_{2^{\alpha-i}} \otimes \left(H_2H_2^H\right) \otimes \left(I_{2^{i-1}}I_{2^{i-1}}^H\right) \\ &= 2(I_{2^{\alpha-i}} \otimes I_2 \otimes I_{2^{i-1}}) = 2I_{2^\alpha}. \end{aligned} \tag{27}$$

View Source

Lemma 2

$w_{F_i}w_{F_i}^H = w_{F_i}^H w_{F_i} = I.$

Proof

$$w_{F_i}w_{F_i}^H = \operatorname{diag}\left(w_{F_{i,j}}w_{F_{i,j}}^*\right) = I_{2^\alpha}$$

$$0 \le i \le \alpha - 1, 0 \le j \le 2$$

View Source

The total power of quantization noise of DIF FFT algorithms, P_{nf}, is calculated by $\operatorname{tr}[E[\Delta X_F \Delta X_F^H]]$ as shown in (29). n_{F_i} is defined as the number of nontrivial complex multiplications at the ith stage of DIF FFT algorithms.

$$\begin{aligned} P_{nf} &= \operatorname{tr}\left[E\left[\Delta X_F \Delta X_F^H\right]\right] \\ &= \operatorname{tr}\left[E\left[\sum_{i=0}^{\alpha-1}\prod_{j=i+1}^{\alpha-1} w_{F_j}B_{\alpha,\alpha-j}e_i \right.\right. \\ &\qquad \left.\left. \times \sum_{m=0}^{\alpha-1} e_m^H \prod_{n=m+1}^{\alpha-1} B_{\alpha,\alpha-n}^H w_{F_n}^H\right]\right] \\ &= \operatorname{tr}\left[E\left[\sum_{i=0}^{\alpha-1}\left(\prod_{j=i+1}^{\alpha-1} w_{F_j}B_{\alpha,\alpha-j}\right)\right.\right. \\ &\qquad \left.\left. \times e_i e_i^H \left(\prod_{j=i+1}^{\alpha-1} B_{\alpha,\alpha-j}^H w_{F_j}^H\right)\right]\right] \\ &= E\left[\sum_{i=0}^{\alpha-1}\operatorname{tr}\left[\left(\prod_{j=i+1}^{\alpha-1} w_{F_j}B_{\alpha,\alpha-j}\right)\right.\right. \\ &\qquad \left.\left. \times e_i e_i^H \left(\prod_{j=i+1}^{\alpha-1} B_{\alpha,\alpha-j}^H w_{F_j}^H\right)\right]\right] \\ &= E\left[\sum_{i=0}^{\alpha-1}\operatorname{tr}\left[(e_i e_i^H)\left(\prod_{j=i+1}^{\alpha-1} B_{\alpha,\alpha-j}^H w_{F_j}^H\right)\right.\right. \\ &\qquad \left.\left. \times \left(\prod_{j=i+1}^{\alpha-1} w_{F_j}B_{\alpha,\alpha-j}\right)\right]\right] \\ &= E\left[\sum_{i=0}^{\alpha-1}\operatorname{tr}\left[(e_i e_i^H)\left(\prod_{j=i+1}^{\alpha-1} B_{\alpha,\alpha-j}^H w_{F_j}^H w_{F_j}B_{\alpha,\alpha-j}\right)\right]\right] \\ &= E\left[\sum_{i=0}^{\alpha-1} 2^{\alpha-i-1}\operatorname{tr}\left[e_i e_i^H\right]\right] \\ &= \sum_{i=0}^{\alpha-1} 2^{\alpha-i-1}n_{F_i}\sigma_c^2. \end{aligned} \tag{29}$$

View Source

B. Propagation Model of DIT FFT Algorithms

Similarly, as shown in Fig. 3, we can define ΔX_T as the overall output error of DIT FFT algorithms:

$$\Delta X_T = B_{\alpha,\alpha} \sum_{i=0}^{\alpha-1} \left(\prod_{j=i+1}^{\alpha-1} w_{T_j} B_{\alpha,j} \right) e_i \tag{30}$$

View Source

where w_{T_i} is the equivalent twiddle factor matrix at the ith stage of DIT FFT algorithm. Therefore, the total power of quantization noise of DIT FFT algorithm, P_{nt}, can be calculated as follows:

$$\begin{aligned}
P_{nt} &= \mathrm{tr}\left[E\left[\Delta X_T \Delta X_T^H\right]\right] \\
&= \mathrm{tr}\left[E\left[B_{\alpha,\alpha} \sum_{i=0}^{\alpha-1} \left(\prod_{j=i+1}^{\alpha-1} w_{T_j} B_{\alpha,j} \right) \right.\right. \\
&\qquad \left.\left. \times e_i \sum_{m=0}^{\alpha-1} e_m^H \left(\prod_{n=m+1}^{\alpha-1} B_{\alpha,n}^H w_{T_n}^H \right) B_{\alpha,\alpha}^H \right]\right] \\
&= \mathrm{tr}\left[E\left[B_{\alpha,\alpha} \sum_{i=0}^{\alpha-1} \left(\prod_{j=i+1}^{\alpha-1} w_{T_j} B_{\alpha,j} \right) \right.\right. \\
&\qquad \left.\left. \times e_i e_i^H \left(\prod_{j=i+1}^{\alpha-1} B_{\alpha,n}^H w_{T_j}^H \right) B_{\alpha,\alpha}^H \right]\right] \\
&= E\left[\mathrm{tr}\left[\sum_{i=0}^{\alpha-1} \left(\prod_{j=i+1}^{\alpha-1} w_{T_j} B_{\alpha,j} \right) \right.\right. \\
&\qquad \left.\left. \times e_i e_i^H \left(\prod_{j=i+1}^{\alpha-1} B_{\alpha,n}^H w_{T_j}^H \right) B_{\alpha,\alpha}^H B_{\alpha,\alpha} \right]\right] \\
&= E\left[2 \cdot \sum_{i=0}^{\alpha-1} \mathrm{tr}\left[\left(e_i e_i^H\right) \left(\prod_{j=i+1}^{\alpha-1} B_{\alpha,n}^H w_{T_j}^H \right) \right.\right. \\
&\qquad \left.\left. \times \left(\prod_{j=i+1}^{\alpha-1} w_{T_j} B_{\alpha,j} \right) \right]\right] \\
&= E\left[2 \cdot \sum_{i=0}^{\alpha-1} \mathrm{tr}\left[\left(e_i e_i^H\right)\left(\prod_{j=i+1}^{\alpha-1} B_{\alpha,\alpha-j}^H w_{T_j}^H w_{T_j} B_{\alpha,\alpha-j} \right)\right]\right] \\
&= E\left[\sum_{i=0}^{\alpha-1} 2^{\alpha-i} tr\left[e_i e_i^H\right]\right] \\
&= \sum_{i=0}^{\alpha-1} 2^{\alpha-i} n_{T_i} \sigma_c^2.
\end{aligned} \tag{31}$$

View Source

Here, we define n_{T_i} as the number of nontrivial twiddle factors at the ith stage. The signal flow graph of DIT algorithm is the transpose of the signal flow graph of the DIF algorithm and we can also present the relationship between of w_{F_i} and w_{T_i} as follows:

$$w_{F_i} = w_{T_{\alpha-i-1}}, \qquad 0 \le i \le \alpha - 1. \tag{32}$$

View Source

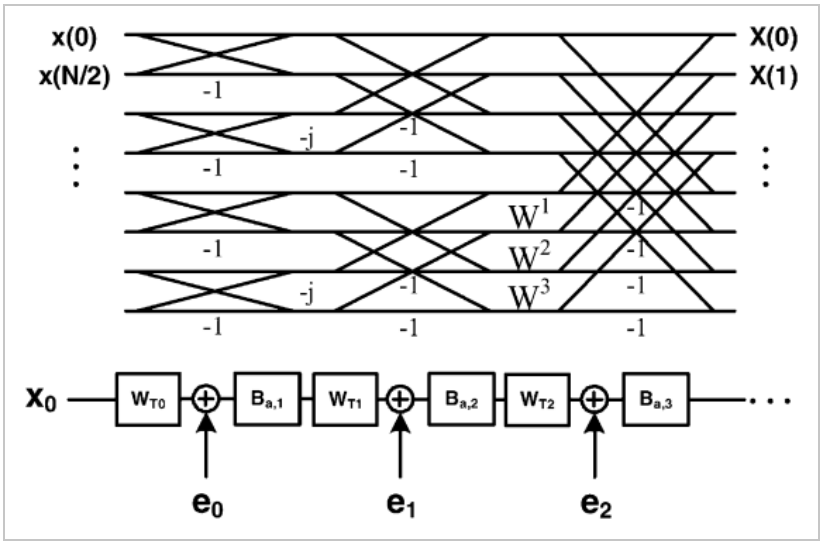

Fig. 3

Fig. 3.
Propagation model of quantization loss for DIT FFT.

From (29) and (31), we observe that the butterfly matrices $B_{\alpha,i}$ act as an internal amplifier that doubles the noise power after every stage. Therefore, if most of nontrivial twiddle factors are located in the later stages of FFT operation, the overall power of quantization loss will be smaller. Besides, we can also compare the overall quantization noise power of different FFT algorithms by finding the number of nontrivial twiddle factors, n_{F_i} or n_{T_i}, at each stage. In order to make a quantitative comparison, without loss of generality, we assume all the quantization noises are mutually independent uniform random variables with variance σ_c^2, then the overall noise power of each FFT algorithms with alternative decimation schemes is listed in Tables II and III. Again, if the length of FFT operations is not power of 4, we use a cascaded radix-2 stage in the last stage for radix-4 DIF algorithms and a leading radix-2 stage for radix-4 DIT algorithms. Based on the observation from Tables II and III, because most of nontrival twiddle factors are concentrated in the later stages, radix-2 DIT FFT has the best signal-to-quantization-noise ratio (SQNR) performance if no scaling operation is performed, which agrees with the conclusion of [17]. Generally speaking, the DIT representation of Cooly–Tukey style FFT algorithms such as radix-2 FFT and radix-4 FFT outperforms its own DIF version. However, due to the irregular twiddle factor structure of split-radix FFT algorithms, a comprehensive simulation is performed to investigate its effect. As listed in Tables II and III, DIF split-radix FFT is better than DIT split-radix FFT. The corresponding simulation results will be discussed in Section IV.

C. Optimal Bit Allocation

According to (29) and (31), we can improve the SQNR performance by minimizing the quantization loss that arose in the early stages of FFT operations if the wordlength of internal memory cells is not fixed for all stages. As for the implementation of FFT algorithms, most previous literature addresses two design issues: the efficiency of complex multipliers and the memory strategy. The design parameters have to be adjusted according to different target applications. Designers focus on increasing the utilization rate of multipliers in the design of short-length FFT architecture. Conversely, the memory access strategy becomes more important in designing a long-length FFT architecture.

First, we discuss the case of short-length FFT implementation. When considering the hardware design of short-length FFT architectures, delay feedback (DF) memory allocation is mostly used due to its higher utilization rate compared to that of the delay commutator (DC) scheme [8], [22]. The main idea of DF scheme is to store the first half of the input samples in the DF memory cells and wait until the second half of input samples arrive. Although the total count of memory cells is the same for both types of FFT algorithms as expressed in (33) and (34), the number of memory cells of DIT algorithms will be doubled after every butterfly stage while that of the DIF algorithms will be half after each stage as shown in Fig. 4.

$$\text{DIF}: M_F = \sum_{i=0}^{\alpha-1} 2^{\alpha-i-1} = 2^{\alpha} - 1 \tag{33}$$

$$\text{DIT}: M_T = \sum_{i=0}^{\alpha-1} 2^{i} = 2^{\alpha} - 1. \tag{34}$$

View Source

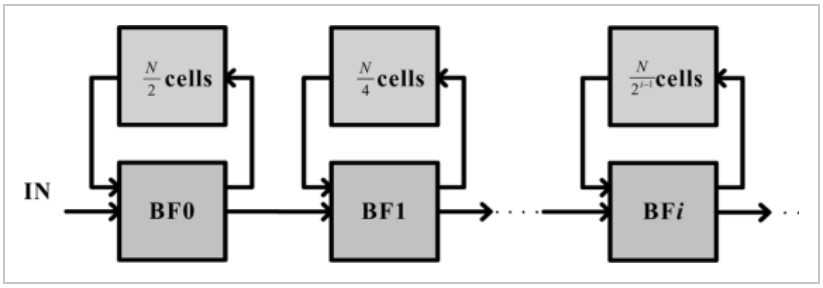

Fig. 4.
Block diagram of DF memory scheme for DIF FFT.

Given the same input signals, we minimize the quantization loss caused by internal arithmetic rounding operations to improve the overall SQNR performance by adjusting the wordlength of intermediate results. The Lagrange multiplier is a commonly adopted mathematical tool to find the local extreme of a multivariable function subject to one or many constraint functions. Extra dummy variables, $\lambda_i (0 \le i \le k-1)$, are introduced to convert the original problem set with n variables and k constraint functions to a dual problem set with $n + k$ variables and no constraint function. Recall that the variance of complex multipliers σ_c^2 can be represented by the MSE of real multiplications σ_r^2. Without loss of generality, σ_c^2, σ_r^2, and internal bit length b can be related in (35) for different

b^{-}. Without loss of generality, b_c, b^{-}, and internal bit length b_c^{-} can be related in (35) for different implementations of complex multipliers, where $3 \leq \beta \leq 4$ [20], as follows:

$$\sigma_c^2 = \beta\sigma^2 = \beta\frac{2^{-2b_c}}{12}. \tag{35}$$

View Source

For the DIT FFT implementations with DF memory scheme, the total number of memory cell B_T is expressed as

$$B_T = 2 \cdot \sum_{i=0}^{\alpha-1} 2^i (b_i + b_x) \tag{36}$$

View Source

where b_x represents the effective wordlength of input samples and can be regarded as the minimal wordlength for internal memory storage. Besides, b_i is number of extra bits used in the ith stage of FFT computation.

To apply the Lagrange multiplier optimization, the objective function is shown in (31) and the constraint function is the overall bit budget expressed in (36). Therefore, the bit allocation can be represented as an optimization problem formulated as follows:

$$\begin{aligned} P_{nt}(b_i, \lambda_T) = &\sum_{i=0}^{\alpha-1} 2^{\alpha-i} n_{T_i} \beta \frac{2^{-2(b_i+b_x)}}{12} \\ &+\lambda_T \left(2 \cdot \sum_{i=0}^{\alpha-1} 2^i (b_i + b_x) - B_T \right). \end{aligned} \tag{37}$$

View Source

By taking partial derivative on (37) with respect to b_i, we can calculate b_i in terms of λ_T as follows:

$$\frac{\partial P_{nt}}{\partial b_i} = 2^{\alpha-i} n_{T_i} \beta \frac{2^{-2(b_i+b_x)}}{12} \ln 2(-2) + 2^{i+1}\lambda_T \equiv 0 \tag{38}$$

$$b_i = \frac{1}{2}\left(\alpha - 2i - 1 + \log_2 n_{T_i}\beta\frac{\ln 2}{6} - \log_2 \lambda_T\right) - b_x. \tag{39}$$

View Source

Also, by taking partial derivative on (37) with respect to λ_T, it yields

$$\frac{\partial P_{nt}}{\partial \lambda_T} = 2 \cdot \sum_{i=0}^{\alpha-1} 2^i (b_i + b_x) - B_T \equiv 0. \tag{40}$$

View Source

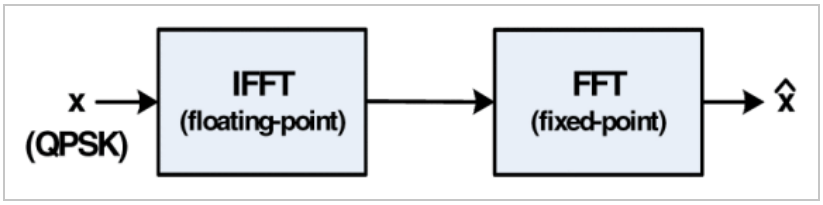

Fig. 5.
Fixed-point simulation with QPSK signals.

Plugging (39) into (40), we can calculate λ_T in (41) and (42), shown at the bottom of the next page. Then, we plug (42) into (39) to find the expression of b_i as formulated in (43), shown at the bottom of the next page.

Similarly, for the DIF FFT implementations with DF memory scheme, the total number of bits used for memory cells B_F is expressed as

$$B_F = 2 \cdot \sum_{i=0}^{\alpha-1} 2^{\alpha-i-1} (b_i + b_x). \tag{44}$$

View Source

In order to find the optimal bit allocation, we need to solve the Lagrange equation expressed in (45), where the objective function is calculated in (29) and the constraint function is the number of overall bit budget as shown in (44).

$$P_{nf}(b_i, \lambda_F) = \sum_{i=0}^{\alpha-1} 2^{\alpha-i-1} n_{F_i} \beta \frac{2^{-2(b_i+b_x)}}{12} + \lambda_F \left(2 \cdot \sum_{i=0}^{\alpha-1} 2^{\alpha-i-1} b_i - B_F\right) \tag{45}$$

View Source

First of all, by taking partial derivative on (45) with respect to b_i, we can express b_i in term of λ_F as

$$\frac{\partial P_{nf}}{\partial b_i} = 2^{\alpha-i-1} n_{F_i} \beta \frac{2^{-2(b_i+b_x)}}{12} \ln 2(-2) + 2^{\alpha-i} \lambda_F \equiv 0 \tag{46}$$

$$b_i = \frac{1}{2}\left(\log_2 n_{F_i} \beta \frac{\ln 2}{6} - 1 - \log_2 \lambda_F\right) - b_x. \tag{47}$$

View Source

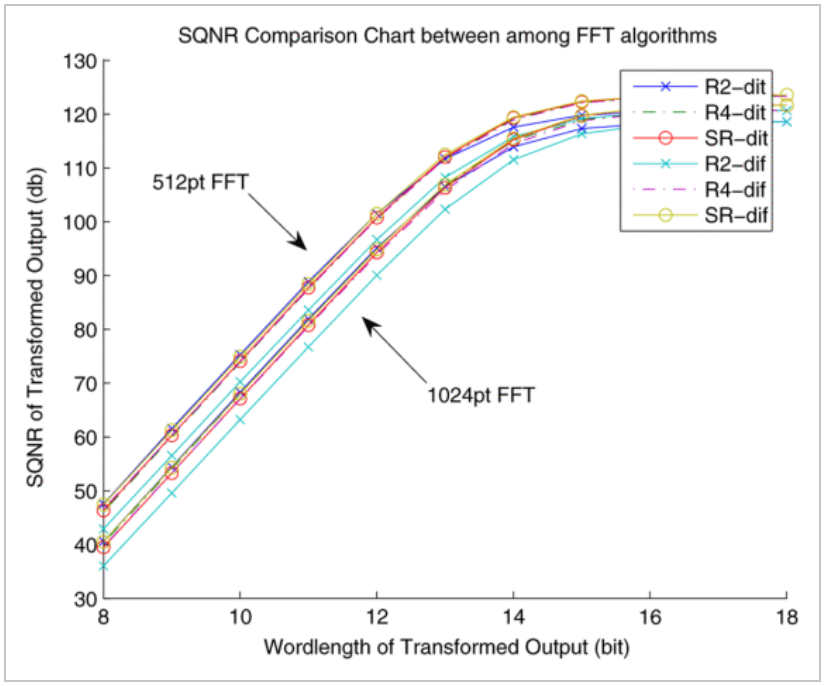

Fig. 6.
SQNR comparison chart among FFT algorithms with fixed-point arithmetic.

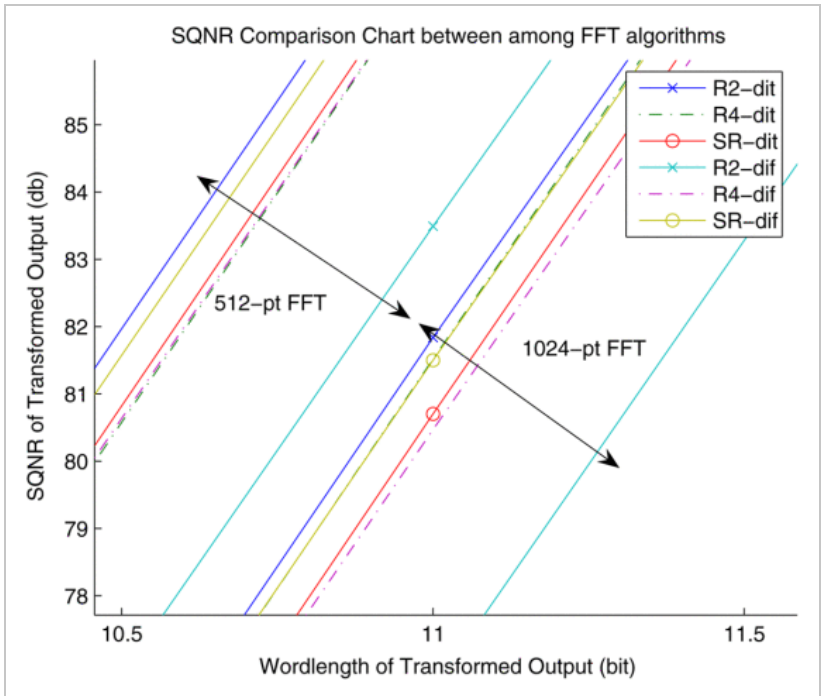

Fig. 7.
Zoom-in version of SQNR comparison chart among FFT algorithms with fixed-point arithmetic.

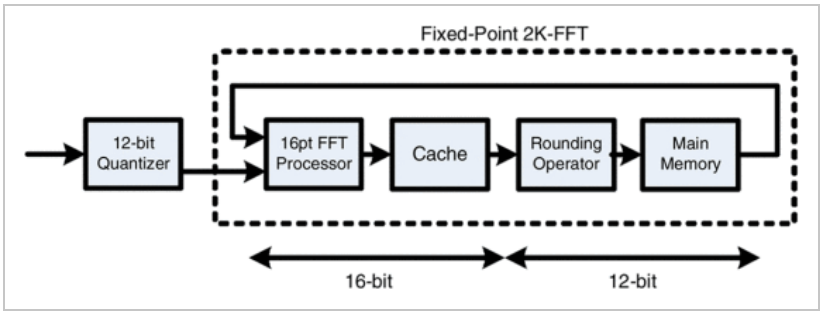

Fig. 8.
Block diagram of 2048-point fixed-point FFT simulation.

Second, we can take the partial derivative on (45) with respect to λ_F as

$$\frac{\partial P_{nf}}{\partial \lambda_F} = 2 \cdot \sum_{i=0}^{\alpha-1} 2^{\alpha-i-1}(b_i + b_x) - B_F \equiv 0. \tag{48}$$

View Source

Plug (47) into (48), we can calculate λ_F in (49) and (50).

$$\begin{aligned}
\frac{\partial P_{nt}}{\partial \lambda_T} &= 2 \cdot \sum_{i=0}^{\alpha-1} 2^{i}(b_i + b_x) - B_T \\
&= \left(\alpha - 1 + \log_2 \beta \frac{\ln 2}{6} - \log_2 \lambda_T\right)(2^{\alpha} - 1) + \sum_{i=0}^{\alpha-1} 2^{i} \log_2 n_{T_i} - (\alpha - 2)2^{\alpha+1} - 4 - B_T \\
\log_2 \lambda_T &= \frac{1}{2^{\alpha} - 1}\left[\sum_{i=0}^{\alpha-1} 2^{i} \log_2 n_{T_i} - (\alpha - 2)2^{\alpha+1} - 4 - B_T\right] + \left(\alpha - 1 + \log_2 \beta \frac{\ln 2}{6}\right) \\
b_i &= \underbrace{\frac{1}{2}(\log_2 n_{T_i} - 2i)}_{\text{variables}} + \underbrace{\frac{1}{2(2^{\alpha} - 1)}\left((\alpha - 2)2^{\alpha+1} + 4 + B_T - \sum_{i=0}^{\alpha-1} 2^{i} \log_2 n_{T_i}\right) - b_x}_{\text{constants}}
\end{aligned}$$

View Source

Therefore, by inserting (50) into (47), we can find the expression of b_i for DIF FFT algorithms formulated as follows:

$$\begin{aligned}
&\frac{\partial P_{nf}}{\partial \lambda_F} \\
&\quad = \sum_{i=0}^{\alpha-1} 2^{\alpha-i-1}(b_i + b_x) - B_F \\
&\quad = \left(\log_2 \beta \frac{\ln 2}{6} - 1 - \log_2 \lambda_F\right)(2^{\alpha} - 1) \\
&\qquad + \sum_{i=0}^{\alpha-1} 2^{\alpha-i-1} \log_2 n_{F_i} - B_F
\end{aligned} \tag{49}$$

$$\begin{aligned}
&\log_2 \lambda_F \\
&\quad = \left(\log_2 \beta \frac{\ln 2}{6}\right) - 1 \\
&\qquad + \frac{1}{(2^{\alpha} - 1)}\left(\sum_{i=0}^{\alpha-1} 2^{\alpha-i-1} \log_2 n_{F_i} - B_F\right)
\end{aligned} \tag{50}$$

$$b_i = \frac{1}{2}\left[\underbrace{\log_2 n_{F_i}}_{\text{variables}} + \underbrace{\frac{1}{2^{\alpha} - 1}\left(B_F - \sum_{i=0}^{\alpha-1} 2^{\alpha-i-1} \log_2 n_{F_i}\right)}_{\text{constants}}\right] - b_x. \tag{51}$$

View Source

SECTION IV.
Performance Analysis

A. Quantitative Simulation Results

In order to verify the expressions derived in the previous section, a fixed-point simulation environment is implemented as shown in Fig. 5. First, floating-point inverse FFT (IFFT) is performed with random QPSK signals. The wordlength of twiddle factors coefficients is also set to 10 bits. The internal wordlength of fixed-point FFT is swept from 8 bits to 18 bits. Both 512-point and 1024-point FFTs are simulated. As seen from Fig. 6, the SQNR value of all 1024-point simulations is lower than that of 512-point simulations because the internal arithmetic rounding errors will accumulate as the length of FFT increases. Based on observation of Fig. 7, radix-2 DIT algorithms have the best accuracy of all. For both radix-2 and radix-4 FFT algorithms, the DIT versions have better SQNR performance than the DIF versions, which agrees with the general rules described in the previous section. However, if split-radix FFT algorithm is adopted, the DIF version will outperform its own DIT version, which is as expected from Tables II and III. It is also clear from Fig. 7 that all decompositions except radix-2 DIF FFT have comparative performance.

B. Bit Allocation of Short-Length FFT Architecture

In order to compare the accuracy of fixed-point FFTs with different internal wordlength configurations, three different settings of internal wordlength are specified. The simulation flow is described as follows. First of all, a 128-point floating-point IFFT is performed with QPSK signals. The transformed output of IFFT computation is fed into an 8-bit quantizer then passed to a fixed-point DIT split-radix FFT to calculate the SQNR performance. The number of the first row, Case (Inc), listed in Table IV shows the results from increasing 1 bit after every BF stage. The second row, Case (Same), shows the results when all intermediate results are stored in 11-bit memory cells. The third row shows the result of memory scheme recommended from the optimization equation derived in the last section. A total of 10 000 trials are run. As observed from Table V, blindly increasing the internal wordlength by 1 bit will use most memory usage and lead to worst performance compared to the other two configurations. For the comparison between the second case and the third case, it is observed that the optimized wordlength setting yields comparative simulation results with fewer bit counts.

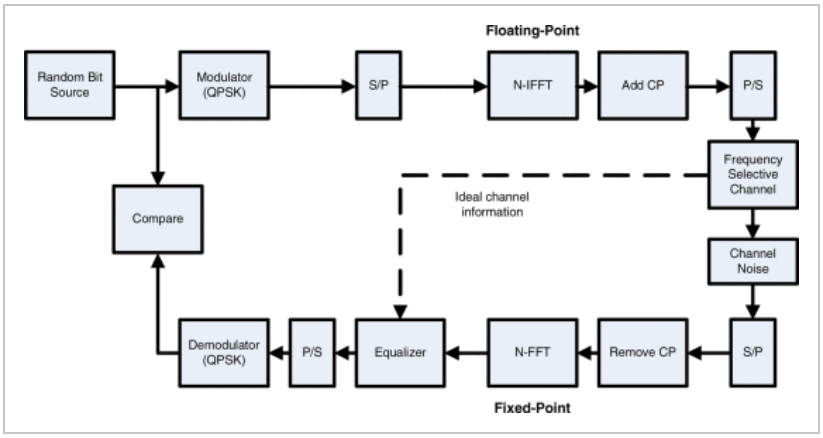

Fig. 9.
Block diagram of 2048-point fixed-point FFT simulation with frequency-selective fading channel.

C. Simulation with the Presence of Noisy Channel

The main bottleneck of DF-based architecture is the number of concurrent memory access increases as the length of FFT increases. This leads to the requirement of multiple separated memory blocks and duplicate butterfly modules. Therefore, for the design of long-length FFT architecture, both the accuracy of transformed output and the feasible memory access scheme become important issues. Several previous research have addressed these two design obstacles. Choi et al. [26] proposed the coverage block floating point (CBFP) method to dynamically normalize the intermediate results after each stages. In [27], the authors also utilize the idea of CBFP with an on-the-fly normalization module so as to remove the usage of temporary buffers. In [28] and [29], the authors increased the efficiency of memory access by inserting two caches and reported that the memory blocks are the largest blocks for long-length FFT architecture. Generally speaking, the designs of long-length FFT architecture uses centralized memory blocks to save hardware cost. As a result, we cannot freely adjust the wordlength of intermediate memory storages at every stage. Recall from (29) and (31), we can reduce the overall noise power by assigning more bits in the early stage of FFT computation. Therefore, in order to achieve better accuracy, a 2048-point split-radix DIF FFT architecture with improved FFT processor is simulated.

As shown in Fig. 8, the wordlength of the main memory block is 12 bits and that of FFT core and internal cache is set to 16 bits. Scaling operations are performed after every butterfly stage to maintain the dynamic range. A simulation environment with frequency selective channel is implemented to observe the effect of bit error rate (BER) with improved FFT processor. The channel is modeled as a finite-length FIR filter as a quasi-static model. It has three consecutive paths which are located at integer chip position. Their relative power profile is [0 0 0] dB, which is normalized so that the overall average channel impulse response energy is equal to 1. The noise at the receiver is assumed to be additive white complex Gaussian noise. The SNR is defined as the ratio of the average signal tone power to the noise power at a frequency bin. In order to estimate the average BER, at least 1000 channel realizations are simulated for each SNR point The receiver is assumed to have perfect knowledge of the channel state information. The signal flow is shown in Fig. 9. The channel SNR is swept from 0 to 40 dB. As we can see from Fig. 10, the long-length FFT computation with improved FFT processor (listed as "fxp op") can achieve better BER under high-SNR conditions.

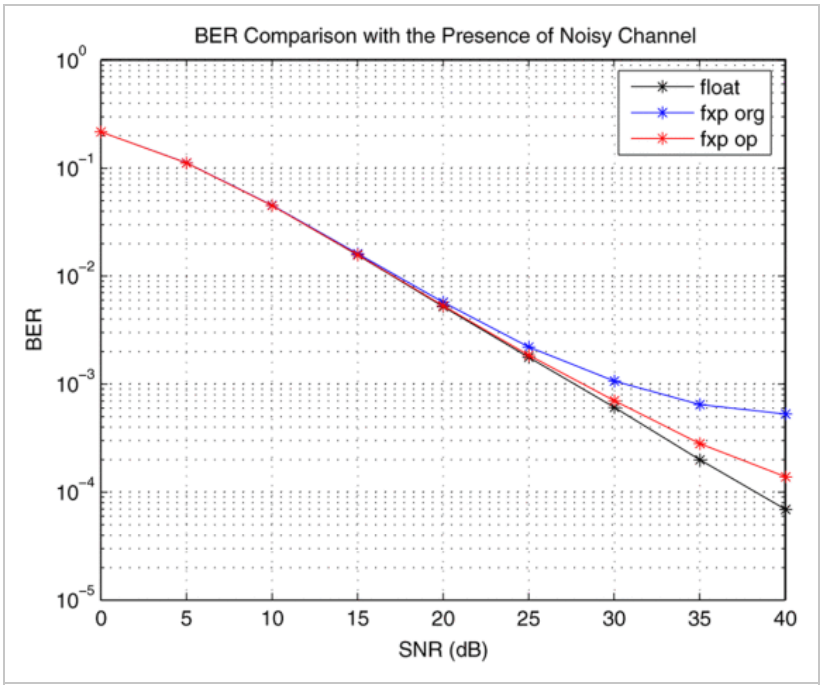

Fig. 10.
Bit error rate comparison with improved FFT processor.

Table II Comparison of Overall Noise Power for DIF FFT (Unit: σ_c^2)

FFT Size	Radix-2	Radix-4	Split-Radix
8	8	4	4
16	64	32	28
32	352	176	148
64	1664	832	684
128	7296	3648	2964
256	30720	15360	12396
512	126464	63232	50836
1024	514048	257024	206188

Table III Comparison of Overall Noise Power for DIT FFT (Unit: σ_c^2)

FFT Size	Radix-2	Radix-4	Split-Radix
8	4	8	8
16	28	32	40

32	140	208	200
64	620	688	840
128	2604	3696	3528
256	10688	11760	14280
512	43180	60656	57800
1024	173740	191216	231880

Table IV Wordlength of Each Intermediate Memory Stages

	Wordlength (bits)						
	1st	2nd	3rd	4th	5th	6th	7th
Case (Inc)	8	9	10	11	12	13	14
Case (Same)	11	11	11	11	11	11	11
Case (Opt)	14	15	15	15	12	12	9

Table V Comparison of Different Memory Scheme, 128-Point DIT SR-FFT

	SQNR (dB)	Change(%)	Bit Count	Change(%)
Case (Inc)	73.1031	0.00%	3316	0.00%
Case (Same)	73.4597	0.49%	2794	-15.74%
Case (Opt)	73.4853	0.52%	2752	-17.01%

SECTION V. Conclusion

In this paper, a comprehensive study is presented for the effect of fixed-point arithmetic in FFT operations. We have derived an analytic expressions for the noise power of overall arithmetic rounding errors for all FFT algorithms. The theoretical derivations show the significance of the location of the nontrivial complex multiplications. Because the quantization noise power will be amplified after each butterfly stage, the less introduced noise power in the early stage will lead to better performance. Compared to previous works, a general propagation model is proposed to quickly estimate the arithmetic quantization errors of different FFT algorithms. From the simulation results most FFT algorithms yield similar performance except radix-2 DIF FFT. An operational optimization procedure is also proposed to reduce the hardware cost without sacrificing SQNR performance for short-length FFT architecture. From the simulation results, the FFT modules with higher accuracy will lead to better BER with the presence of noisy channel. Further research could be done to study the effect of fixed-point arithmetic for long-length FFT architecture as both scaling operation and block floating point arithmetic are used.

ACKNOWLEDGMENT

The authors would like to thank their colleagues M.-P. Kao, M. Li, and K. S. C. Pun for many substantial discussions. The authors would also like to thank Texas Instruments, Inc., for their financial support and the anonymous reviewers for their constructive comments.