ERROR ANALYSIS AND COMPLEXITY OPTIMIZATION FOR THE MULTIPLIER-LESS FFT-LIKE TRANSFORMATION (ML-FFT)

K. M. Tsui, S. C. Chan and K. W. Tse

Department of Electrical and Electronic Engineering,
The University of Hong Kong, Pokfulam Road, Hong Kong.

ABSTRACT

This paper studies the effect of the signal round-off errors on the accuracies of the multiplier-less Fast Fourier Transform-like transformation (ML-FFT). The idea of the ML-FFT is to parameterize the twiddle factors in the conventional FFT algorithm as certain rotation-like matrices and approximate the associated parameters inside these matrices by the sum-of-power-of-two (SOPOT) or canonical signed digits (CSD) representations. The error due to the SOPOT approximation is called the coefficient round-off error. Apart from this error, signal round-off error also occurs because of insufficient wordlengths. Using a recursive noise model of these errors, the minimum hardware to realize the ML-FFT subject to the prescribed output bit accuracy can be obtained using a random search algorithm. A design example is given to demonstrate the effectiveness of the proposed approach.

I. INTRODUCTION

The Discrete Fourier Transform (DFT) is an important tool in digital signal processing [1]. A treasure of fast algorithms such as the Cooley-Tukey Fast Fourier Transform (FFT) and the prime factor algorithm (PFA) FFT are available to compute efficiently DFT of different lengths. Recently, the efficient realization of the multiplier-less FFT based on the integer [2,3] or SOPOT representation [4], and its extension to the multiplier-less sinusoidal transforms [5] have been proposed. The main objective is to avoid the expensive general-purpose multipliers which are replaced with limited number of shifters and adders. However, this approximation will unavoidably introduce errors which are referred to the coefficient round-off errors. Fortunately, as proposed in [4], tradeoffs between the arithmetic complexities and the output accuracies can be made so that the minimum arithmetic complexities can be obtained for different applications, which require different degree of the error tolerance. Due to finite wordlength of internal representation, another source of error, called signal round-off error [1], occurs when rounding is performed for the intermediate data after complex multiplication with the twiddle factor. Moreover, overflow can occur due to insufficient internal wordlength when fixed-point arithmetic is used. Unfortunately, most design methods for the multiplier-less FFT only focus on the effect of the coefficient round-off errors. In order to satisfy the prescribed output accuracy, one usually employs fixed but rather long wordlength for all intermediate data, which means increased hardware complexity. Therefore, it is necessary to design a general model to determine the minimum hardware complexity, subject to a given output accuracy.

In this paper, we propose a new recursive round-off noise model for computing the output bit accuracies of the ML-FFT under finite wordlength effect. Without loss of generality, the decimation-in-time (DIT) radix-p ML-FFT is used as an example. The noise sources due to the rounding operations performed after multiplications are first identified at each stage, based on the structure of the DIT radix-p FFT, where the size of the transformation N is the integer power of p. For each output point at any stage, its noise powers are determined statistically by its associated noise sources, using the commonly used uncorrelated white noise model. Together with the noise powers coming from the previous stage, the total noise powers of all the output points can be calculated, and propagate to the next stage. Eventually, the final output bit accuracy of each output point can be obtained by summing the total noise powers accumulated at this output point. Using these results, the internal wordlength of each intermediate data can then be optimized subject to prescribed output accuracy using a random search algorithm [8,9]. As an illustration, the number of adder cells and registers used, which is related to the exact wordlength used for each intermediate data, is chosen as a measure of the hardware complexity. Design result shows that our proposed approach can efficiently determine the minimum hardware complexity subject to prescribed output bit accuracy. The rest of this paper is organized as follows: Section II describes the ML-FFT algorithm based on the DIT radix-p FFT. Section III is devoted to the error analysis of the ML-FFT and the wordlength determination method. A design example demonstrating the effectiveness of the proposed approach is given in Section VI. Finally, conclusion is drawn in Section VII.

II. THE ML-FFT ALGORITHM

A. — The decimation-in-time (DIT) radix-p FFT

The discrete Fourier transform (DFT) of an N-point sequence {x(n)} is given by:

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn} \quad \text{for } k = 0,1,...,N-1, \tag{2-1}$$

where $W_N = e^{-i(2\pi/N)}$ and $i = \sqrt{-1}$. Consider the general formula of the DIT radix-p FFT as follow:

$$X[k + r(N/p)] = \sum_{n=0}^{N/p-1} \left(\sum_{j=0}^{p-1} x[pn + j] \cdot W_p^{jr} W_N^{jk} \right) W_{N/p}^{nk} \tag{2-2}$$

for $k = 0,1,...,N/p-1$ and $r = 0,1,...,p-1$. Figure 1 shows the block diagram of the DIT radix-p FFT for computing a DFT of length $N = p^m$. Using the above decomposition, the DFT can be reduced successively to N/p p-point DFTs. In general, this process can be repeated m times and therefore there are totally m stages in the implementation of the DFT.

B. — Mulitplier-less realization

The ML-FFT algorithm [4] approximates the twiddle factor multiplications W_N^{jk} and W_p^{jr} in (2-2) by representing the coefficients of a certain factorization of the rotation using SOPOT coefficients. To start with, let $c = (x + i \cdot y)$ be any complex number. The multiplication of c with $\exp(-i\theta)$, $p = c \cdot \exp(-i\theta)$ can be written as:

$$\begin{bmatrix} p_r \\ p_i \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \boldsymbol{R}_\theta \cdot \begin{bmatrix} x \\ y \end{bmatrix}, \tag{2-3}$$

where $\boldsymbol{R}_\theta = \begin{bmatrix} \cos\theta & \sin\theta \\ \sin\theta & -\cos\theta \end{bmatrix}$ is the rotation-like matrix and $\theta = 2\pi jk/N$ for W_N^{jk} or $\theta = 2\pi jr/p$ for W_p^{jr}. However, $\cos\theta$ and $\sin\theta$ in (2-3) are not suitable for directly conversion to the SOPOT representation since the inverse of $\boldsymbol{R}_\theta$ cannot in general be expressed in terms of SOPOT coefficients. To cope with this, $\boldsymbol{R}_\theta$ is re-written as follows:

$$\boldsymbol{R}_\theta = \begin{bmatrix} 1 & -\tan(\theta/2) \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ \sin\theta & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & \tan(\theta/2) \\ 0 & -1 \end{bmatrix} = \boldsymbol{R}_\theta^{-1}. \tag{2-4}$$

Since the forward and inverse of the matrices involve the same set of coefficients, i.e. $\sin\theta$ and $\tan(\theta/2)$. They can be quantized to the SOPOT coefficients to from

$$\hat{\boldsymbol{R}}_\theta = \begin{bmatrix} 1 & -\beta_\theta \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \alpha_\theta & 1 \end{bmatrix} \begin{bmatrix} 1 & \beta_\theta \\ 0 & -1 \end{bmatrix} \approx \boldsymbol{R}_\theta, \tag{2-5}$$

where α_θ and β_θ are respectively the SOPOT approximations to $\sin\theta$ and $\tan(\theta/2)$ having the form:

$$z_\theta = \sum_{k=1}^{t} a_k 2^{b_k}, \tag{2-6}$$

where $a_k \in \{-1,1\}$, $b_k \in \{-r,\ldots,-1,0,1,\ldots r\}$; r is the range of the coefficients and t is the number of terms used in each coefficient. Using these results, the number of SOPOT terms can then be optimized using the random search algorithm [7] subject to the specified errors between the candidate transform and its ideal counterpart. These errors due to the SOPOT approximation are called coefficient round-off errors, which can be reduced by using more SOPOT terms. Interested readers can refer to [4] for more details. In next section, we shall present the analysis of another noise source called signal round-off error, which will also affect the output accuracy of the ML-FFT.

III. ROUND-OFF ANALYSIS OF THE ML-FFT

A. — Signal round-off analysis

Signal round-off errors occur due to rounding of the intermediate signal after multiplications. Since the exact round-off errors are difficult to analyze, they are usually treated as uncorrelated white noises. For rounding operation, the quantization noise will have zero mean with variance σ equal to $\Delta^2/12$, where Δ is the quantization step-size. In other words, the variance is determined by the number of fractional bits that is retained after multiplication. In fixed-point arithmetic, each intermediate signal can be represented in the form of $<I/F>$, where I is the number of integer bits including the sign bit and F is the number of fractional bits. In general, if F bits are rounded to B bits, where $B<F$, then the noise variance will be given by:

$$\sigma = \Delta^2/12 = 2^{-2(B-1)}/12. \tag{3-1}$$

Without loss of generality, consider the computation of an N-point ($N = p^m$) FFT using the DIT radix-p FFT algorithm as shown in the figure 2.

$e_j^{(m-1)}(k)$ for $k = 0,1,\ldots,N/p-1$ and $j = 0,1,\ldots,p-1$ are the signal round-off noises introduced at the k-th output of the j-th (N/p)-point DFT. The superscript (v) indicates the v-th stage of the FFT structure, for v = 0,1,…,m. Hence, v = 0 and v = m correspond to the input and the output of the radix-p FFT respectively. That is $x(a) = X^{(0)}(a)$ and $X(a) = X^{(m)}(a)$ for $a = 0,1,\ldots,N-1$. In general, $e_j^{(m-1)}(k)$ are white and identically distributed, by symmetry consideration.

$n_j^{(m)}(k)$'s are the additional signal round-off noises introduced by the multiplication with the "approximated" twiddle factors $\hat{W}_N^{jk}$. They are modeled as zero mean white Gaussian noise with noise power depending on the finite word length after performing the multiplications. It should be noted that details for the other noise sources due to the multiplication with the twiddle factors $\hat{W}_p^{jr}$ are omitted in the figure 2 for simplicity. These errors only exist in the radix-8 or higher radices FFT algorithm. In the radix-2 and radix-4 FFT algorithms, the twiddle factors $\hat{W}_p^{jr}$ are ± 1 or $\pm i$ only, so the DFT does not require any multiplications and there is no rounding error in their implementation. In the rest of this section, p is assumed to be equal to 2 or 4. However, it can easily be generalized to higher radices or split-radix FFT algorithms. Next we will discuss the determination of $n_j^{(m)}(k)$. The round-off noise introduced by the rotation-like matrix $\boldsymbol{R}_\theta$ in (2-4) can be computed as in the figure 3. If rounding is performed after each multiplication, three additive noise sources will be introduced as shown in the figure. Let σ_i^2 be the variances of the noise sources r_i for $i = 0,1,2$. Assume σ_i^2 are uncorrelated, white and zero mean, it follows that the output noise variances are given by:

$$\sigma_{R'}^2 = (1+\sin^2\theta\tan^2(\theta/2))\sigma_0^2 + (\tan^2(\theta/2))\sigma_1^2 + \sigma_2^2 \tag{3-2}$$

$$\sigma_{I'}^2 = (\sin^2\theta)\sigma_0^2 + \sigma_1^2. \tag{3-3}$$

Thus, the noise variance of $n_j^{(m)}(k)$ can be written as follows:

$$VAR\{n_j^{(m)}(k)\} = \sigma_{R'}^2 + \sigma_{I'}^2. \tag{3-4}$$

(3-2) and (3-3) can also apply to the ML-FFT with $\tan(\theta/2)$ and $\sin\theta$ replaced by their SOPOT approximations.

By interchanging the summation signs in (2-2), we can see that the real and imaginary parts of the output signal increase by no more than a factor of p from stage to stage, assuming that both real and imaginary part of the output signal are less than one. Therefore, to avoid overflow of the immediate data, the outputs of the p-point DFT are usually scaled by a factor of 1/p. However, another noise sources $s^{(m)}(q)$, where $q = k + r(N/p)$ due to the scaling is introduced. More precisely, if all the precision in multiplying the SOPOT coefficients are retained and only the real part of the final result is rounded to L_R bits, then the noise source for $\mathrm{Re}(X^{(m)}(q))$ have variance equal to

$$VAR\{\mathrm{Re}(X^{(m)}(q))\} = \Delta^2/12 = 2^{-2(L_R-1)}/12. \tag{3-5a}$$

Similarly, when the imaginary part of the final result is rounded to L_I bits, then the variance of the noise source for $\mathrm{Im}(X^{(m)}(q))$ is given by:

$$VAR\{\mathrm{Im}(X^{(m)}(q))\} = \Delta^2/12 = 2^{-2(L_I-1)}/12. \tag{3-5b}$$

Hence, we have

$$VAR\{s^{(m)}(q)\} = VAR\{\mathrm{Re}(X^{(m)}(q))\} + VAR\{\mathrm{Im}^{(m)}(X(q))\}, \tag{3-6}$$

assuming that the round-off noises are uncorrelated. For other radices, the DFT might introduce additional noise sources, which depend on the exact implementation. Another point worth mentioning is that another scaling factor of N at the final output is needed in order to obtain the correct DFTs, but this matter would not be taken into account for our noise model.

$e_j^{(m-1)}(k)$, $n_j^{(m)}(k)$ and $s^{(m)}(k)$ together constitute signal round-off noise at each output $X(q)$ and is denoted by $e^{(m)}(q)$, which is also zero mean and Gaussian distributed. Assuming that the various noise sources are uncorrelated, the variance of $e^{(m)}(q)$ at the m-th stage is given by:

$$\begin{aligned} VAR\{e^{(m)}(q)\} &= \frac{1}{p^2}\sum_{j=0}^{p-1} VAR\{e_j^{(m-1)}(k)\}\left|\hat{W}_N^{jk}\cdot\hat{W}_p^{jr}\right|^2 \\ &+ \frac{1}{p^2}\sum_{j=0}^{p-1} VAR\{n_j^{(m)}(k)\}\left|\hat{W}_p^{jr}\right|^2 + VAR\{s^{(m)}(q)\} \end{aligned}. \tag{3-7}$$

Further, if we assume $e_j^{(m-1)}(k)$ are identically distributed with variance $VAR\{e^{(m-1)}(k)\}$, then (3-7) simplifies to

$$\begin{aligned} VAR\{e^{(m)}(q)\} &= \frac{VAR\{e^{(m-1)}(k)\}}{p^2}\sum_{j=0}^{p-1}\left|\hat{W}_N^{jk}\cdot\hat{W}_p^{jr}\right|^2 \\ &+ \frac{1}{p^2}\sum_{j=0}^{p-1} VAR\{n_j^{(m)}(k)\}\left|\hat{W}_p^{jr}\right|^2 + VAR\{s^{(m)}(q)\} \end{aligned}. \tag{3-8}$$

Hence, the output accuracy A_q at the q-th output, in terms of the number of fractional bits, is approximately given by:

$$A_q = \left\lfloor 10 \cdot [\log_{10}(VAR\{e^{(m)}(q)\}) + \log_{10} 3] \right\rfloor / 6 \text{ bits}. \qquad (3\text{-}9)$$

Using (3-8), it is possible to recursively compute the noise power at each output of the different stage, given the values of $\hat{W}_N^{jk}$ and $\hat{W}_p^{jr}$, and the noise powers at the previous stage. Note that the noise powers will be accumulated and eventually propagate to the final stage. In order to satisfy the required output accuracy, the noise power at each output should be reduced by increasing the internal wordlengths for the fractional bits at different stages in the FFT structure.

B. **— Overflow handling**

Signal overflows occur when the allocated wordlength of the integer bits is insufficient to handle the increase in the integer bits of the output signal after additions. More bits should be allocated to the integer part of the adder output and the register holding it so as to avoid signal overflow. There are two approaches to deal with this situation. The number of bits in the fractional part can either be retained or decreased, depending on the required output accuracy. Obviously, the latter one will introduce additional round-off noise. To determine whether overflow will occur at a particular adder, a conservative measure is used. In this approach, the addition operates in a way that all the signs of the signals will be ignored. Therefore, the worst-case wordlength format at the adder output can always be found. This can ensure that no overflow will occur at any adder output, at the expense of slightly increased hardware complexity. In the next subsection, we will describe the approach to determine the internal wordlength with prescribed output accuracy.

C. **— Wordlength determination**

For a given output accuracy, the idea of the random search is explored to minimize the hardware complexity [8,9] using the proposed noise model of the FFT algorithm. To start with, an objective function regarding the hardware complexity has to be set up. As an illustration, the number of adder cells and/or registers is employed as a measure of the hardware complexity since it is always the major resources in the hardware environment. Also, it is related to the internal wordlengths for the intermediate signals which are the variables that we want to optimize. Note that other meaningful measures can also be used instead. In general, the determination of the internal wordlength can be done in three steps. First of all, the SOPOT approximation of the twiddle factors is found as discussed in section II-B. Secondly, the format of maximum wordlengths for the intermediate signals, including both real and imaginary parts can be obtained by assuming all the inputs are in their maximum values and the precisions of all the signals are retained. Finally, from the sections III-A and III-B, the noise powers introduced by the rounding operation as well as the output bit accuracies can be statistically calculated, with respect to the proposed wordlengths which are found by the random search algorithm. These proposed wordlengths are stored in the vector δ_f which will be optimized together with the other vectors δ storing all the intermediate signal formats. The one with the minimum number of adder cells, while satisfying the prescribed output accuracy, will be declared as the solution of the problem. More precisely, we can formulate the problem as follows:

$$\min_{(\delta,\delta_f)} C(\delta,\delta_f) \ \ subject\ to\ \ P_{total} \le P_{spec}, \qquad (3\text{-}10)$$

where P_{total} and P_{spec} are respectively the total noise power and the specified output accuracy at the DFT output, and $C(\cdot)$ is the objective function in this problem. There are two ways to speed up the search process. The first one is to identify the symmetries stage by stage because of the fact that all the input signals are assumed to be the maximum so that the number of variables can be largely reduced. For instance, to calculate the wordlengths for all the intermediate signals within a radix-2 64-point FFT, only the first two output points at the first stage are required to examine, rather than all 64 output points. The second one is to make a reasonable initial guess in order to shorten the search time. Higher noise power, say, is allowed at the earlier stage. Thus, shorter wordlength is allocated at the earlier stage such that the overall output bit accuracies at the final stage still satisfy the requirement.

IV. DESIGN EXAMPLE

This example shows the effectiveness of the round-off noise model as proposed in the section III. To start with, let's consider the 64-point radix-2 (i.e. p = 2) FFT with the prescribed accuracy at each output equal to 16. As mentioned earlier, the 2-point DFT can be implemented without any multiplications and its outputs at each stage are scaled by a factor of 1/2. The input sequence $\{x(n)\}$, n = 0,1,…,63 are complex values with both real and imaginary parts having the format of <1/13>. i.e. 14 bits with the maximum value equal to 0.99988. After the wordlength optimization as discussed in section III, the entire FFT structure requires 36070 adder cells. Table 1 and 2 show a summary of results and internal wordlength formats at the 6-th stage of the ML-FFT. The output formats at the remaining stages are omitted due to page limitation. To give an idea of the hardware savings of the proposed structure, a comparison with the structure using fixed wordlength is considered below. For the sake of the comparison, the internal wordlengths for all intermediate signals are fixed to 19 bits. The corresponding number of adder cells is 36936 which is slightly higher than that of the proposed one. For simulation purpose, we use Matlab to model the hardware implementation of the ML-FFT and assume that both structures are free from the coefficient round-off errors. Figure 4 shows the output bit accuracies of the radix-2 64-point ML-FFT using proposed wordlength (solid line) and fixed wordlength (dotted line), taking an average over 10000 random generated binary data with the format of <1/13>. Result shows that our proposed structure meets the required bit accuracy quite well with slight deviation upon the 16-bit accuracy. Also, it shows that the output accuracies of the proposed structure are in general higher than that of the structure using fixed wordlength, except for those at output point 0, 15, 31 and 47. The reason is trivial due to the fact that there is no non-trivial multiplication when tracing the data path associated with those 4 output points from the first stage to the final stage. On the other hand, figure 4 also reveals that the more the multiplications with the twiddle factors, the lower the bit accuracies or the higher the round-off errors for that output point are suffered. As we expected, our proposed approach can efficiently control round-off errors by adjusting the internal wordlengths so that the prescribed output bit accuracies are satisfied without any overflow.

V. CONCLUSION

An error analysis of ML-FFT, using the DIT radix-p FFT as an example with N being an integer power of p, is presented. ML-FFT parameterizes the twiddle factors in the conventional radix-p FFT algorithm as certain rotation-like matrices and approximates the associated parameters by sum-of-power-of-two (SOPOT) or canonical signed digits (CSD) representations. Apart from the error due to the SOPOT approximation, there is another error called signal round-off error which also affects the output bit accuracy of the ML-FFT. A recursive noise model is developed to model the statistics properties of these errors. By using this model, a random search algorithm is proposed to efficiently determine the minimum hardware complexity to realize the ML-FFT subject to the prescribed output bit accuracy. Simulation results show good agreement with the theoretical results.

REFERENCES

[1] A. V. Oppenheim and R. W. Schafer, "Discrete-Time Signal Processing," *Prentice-Hall*, 1989.

[2] S. Oraintara, Y-J Chen and T. Q. Nguyen, "Integer Fast Fourier Transform," *IEEE Trans. Signal Processing*, vol. 50, no. 3, March 2002.

[3] S. C. Pei and J. J. Ding, "The Integer Transforms Analogous to Discrete Trigonometric Transforms," *IEEE Trans. Signal Processing*, vol.48, no. 12, pp. 3345-3364, Dec. 2000.

[4] S. C. Chan and P. M. Yiu, "A Multiplier-less 1-D and 2-D Fast Fourier Transform-like Transformation Using Sum-of-power-of-two (SOPOT) Coefficients," *in Proc. IEEE ISCAS'2002*, vol. 4, pp. 755-758, May 2002.

[5] S. C. Chan and K. M. Tsui, "Multiplier-less Real-valued FFT-like Transformation (ML-RFFT) and Related Real-valued Transformations," in *Proc. ISCAS'2003*, vol. 4, pp. 257-260, May 25-28, 2003.

[6] S. C. Chan, W. Liu and K L. Ho, "Perfect Reconstruction Modulated Filter Banks with Sum-of-power-of-two Coefficients," in *Proc. IEEE ISCAS'2000*, vol. 2, pp. 73-76, May 2000.

[7] S. C. Chan and P. M. Yiu, "Multiplier-less discrete sinusoidal and lapped transforms using sum-of-powers-of-two (SOPOT) coefficients," *in Proc. IEEE ISCAS'2001*, vol. 2, pp. 13-16, May 2001.

[8] K. S. Pun, S. C. Chan and K. L. Ho, "Efficient design of a class of multiplier-less perfect reconstruction two-channel filter banks and wavelets with prescribed output accuracy," *Proceedings of the 11th IEEE Signal Processing Workshop on Statistical Signal Processing*, pp. 599-602, 2001.

[9] S. C. Chan and K. S. Yeung, "On the design and multiplier-less realization of digital IF for software radio receivers with prescribed output accuracy," in *Proc. DSP'2002*, vol. 1, pp. 277-280, 2002.

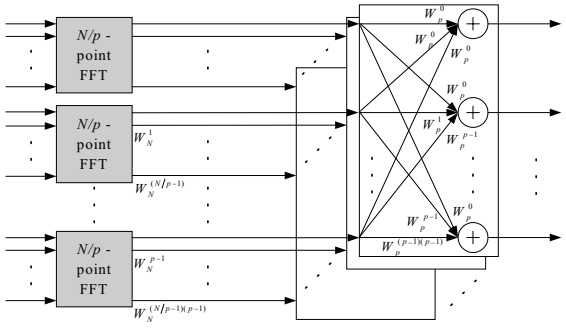


Figure 1: Block diagram of the DIT radix-p N-point FFT, $N = p^m$.

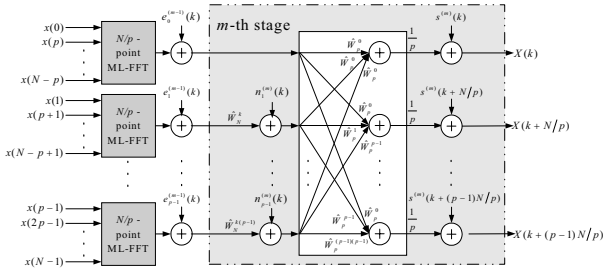


Figure 2: Proposed noise model for the DIT radix-p N-point FFT, $N = p^m$.

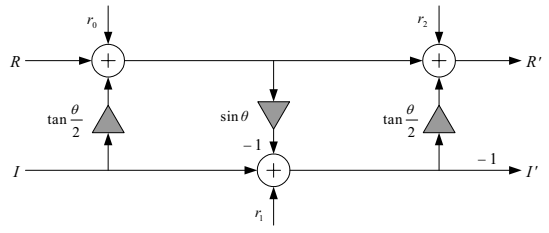
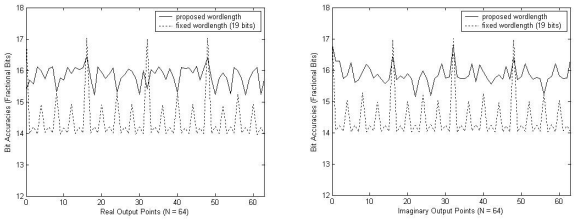


Figure 3: Implementation of the rotation-like matrix (refer to (2-4)) with round-off noises r_i for $i = 0, 1, 2$.



4a) Real Part Storage 4b) Imaginary Part Storage

Figure 4: Output bit accuracies for a) real part storage and b) imaginary part storage of the DIT radix-2 64-point FFT.

TF	Real.	Imag.	TF	Real.	Imag.
W_{64}^{1}	<3/19>	<3/19>	W_{64}^{2}	<3/18>	<3/18>
W_{64}^{3}	<3/19>	<3/19>	W_{64}^{4}	<3/18>	<2/19>
W_{64}^{5}	<4/18>	<3/19>	W_{64}^{6}	<3/18>	<3/18>
W_{64}^{7}	<4/18>	<3/19>	W_{64}^{8}	<3/18>	<2/18>
W_{64}^{9}	<4/18>	<3/19>	W_{64}^{10}	<3/18>	<3/18>
W_{64}^{11}	<4/18>	<3/19>	W_{64}^{12}	<3/18>	<2/19>
W_{64}^{13}	<3/19>	<3/19>	W_{64}^{14}	<3/18>	<3/18>
W_{64}^{15}	<3/19>	<3/19>	W_{64}^{17}	<3/19>	<3/19>
W_{64}^{18}	<3/18>	<3/18>	W_{64}^{19}	<3/19>	<3/19>
W_{64}^{20}	<3/18>	<2/19>	W_{64}^{21}	<4/18>	<3/19>
W_{64}^{22}	<3/18>	<3/18>	W_{64}^{23}	<4/18>	<3/19>
W_{64}^{24}	<3/18>	<2/18>	W_{64}^{25}	<4/18>	<3/19>
W_{64}^{26}	<3/18>	<3/18>	W_{64}^{27}	<4/18>	<3/19>
W_{64}^{28}	<3/18>	<2/19>	W_{64}^{29}	<3/19>	<3/19>
W_{64}^{30}	<3/18>	<3/18>	W_{64}^{31}	<3/19>	<3/19>

Table 1: Proposed wordlengths of the output formats after the non-trivial multiplications at the 6-th stage (TF = Twiddle Factor, Real. = Real Storage and Imag. = Imaginary Storage).

O/P	Real.	Imag.	O/P	Real.	Imag.
0	<1/17>	<1/18>	32	<1/17>	<1/18>
1	<3/18>	<3/19>	33	<3/19>	<3/18>
2	<3/18>	<3/19>	34	<3/19>	<3/18>
3	<3/19>	<3/18>	35	<3/19>	<3/18>
4	<2/18>	<2/18>	36	<2/18>	<2/18>
5	<3/18>	<3/19>	37	<3/18>	<3/19>
6	<3/19>	<3/18>	38	<3/19>	<3/18>
7	<3/19>	<3/18>	39	<3/18>	<3/19>
8	<2/17>	<2/18>	40	<2/17>	<2/18>
9	<3/18>	<3/19>	41	<3/19>	<3/18>
10	<3/18>	<3/19>	42	<3/19>	<3/18>
11	<3/19>	<3/18>	43	<3/19>	<3/18>
12	<2/18>	<2/18>	44	<2/18>	<2/18>
13	<3/19>	<3/18>	45	<3/19>	<3/18>
14	<3/19>	<3/18>	46	<3/18>	<3/19>
15	<3/19>	<3/18>	47	<3/19>	<3/18>
16	<1/18>	<1/18>	48	<1/18>	<1/18>
17	<3/18>	<3/18>	49	<3/18>	<3/18>
18	<3/17>	<3/18>	50	<3/17>	<3/18>
19	<3/19>	<3/18>	51	<3/18>	<3/19>
20	<2/18>	<2/18>	52	<2/18>	<2/18>
21	<3/18>	<3/18>	53	<3/18>	<3/18>
22	<3/18>	<3/17>	54	<3/17>	<3/18>
23	<3/19>	<3/18>	55	<3/19>	<3/18>
24	<2/17>	<2/18>	56	<2/18>	<2/17>
25	<3/18>	<3/18>	57	<3/18>	<3/18>
26	<3/18>	<3/17>	58	<3/17>	<3/18>
27	<3/19>	<3/18>	59	<3/18>	<3/19>
28	<2/18>	<2/18>	60	<2/18>	<2/18>
29	<3/18>	<3/19>	61	<3/19>	<3/18>
30	<3/17>	<3/18>	62	<3/17>	<3/18>
31	<3/19>	<3/18>	63	<3/18>	<3/19>

Table 2: Proposed wordlengths of the output formats at the 6-th stage for the 64-point radix-2 ML-FFT (O/P = Output Point, Real. = Real Storage and Imag. = Imaginary Storage).