<table>
<thead>
<tr>
<th><strong>Title</strong></th>
<th>Error analysis and complexity optimization for the multiplier-less FFT-like transformation (ML-FFT)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Author(s)</strong></td>
<td>Tsui, KM; Chan, SC; Tse, KW</td>
</tr>
<tr>
<td><strong>Citation</strong></td>
<td>Proceedings - IEEE International Symposium On Circuits And Systems, 2004, v. 3, p. III401-III404</td>
</tr>
<tr>
<td><strong>Issued Date</strong></td>
<td>2004</td>
</tr>
<tr>
<td><strong>URL</strong></td>
<td><a href="http://hdl.handle.net/10722/46429">http://hdl.handle.net/10722/46429</a></td>
</tr>
<tr>
<td><strong>Rights</strong></td>
<td>This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.; ©2004 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.</td>
</tr>
</tbody>
</table>
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 unavoidable introduce errors which are referred to the coefficient round-off errors.

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 [6,7]. 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:
\[
X[k] = \sum_{n=0}^{N-1} x(n) W_N^n \quad \text{for} \quad k = 0, 1, ..., N-1,
\]
where \( W_N = e^{-j2\pi/N} \) and \( i = \sqrt{-1} \). Consider the general formula of the DIT radix-p FFT as follow:
\[
X[k+r(N/p)] = \sum_{n=0}^{N/p-1} \sum_{j=0}^{r-1} x[pn+j] W_p^n W_{p^r}^k W_{p^r}^j \quad (2-2)
\]
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 \( N = p^r \). 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. Multiplier-less realization
The ML-FFT algorithm [4] approximates the twiddle factor multiplications \( W_p^k \) and \( W_{p^r}^j \) 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:
\[
\begin{bmatrix}
  p_x \\
  p_y 
\end{bmatrix} = \begin{bmatrix}
  \cos \theta & \sin \theta \\
  -\sin \theta & \cos \theta 
\end{bmatrix} \begin{bmatrix}
  x \\
  y 
\end{bmatrix} = \begin{bmatrix}
  1 & 0 \\
  0 & -1 
\end{bmatrix} R_\theta \begin{bmatrix}
  x \\
  y 
\end{bmatrix} \quad (2-3)
\]
where \( R_\theta = \begin{bmatrix}
  \cos \theta & \sin \theta \\
  \sin \theta & -\cos \theta 
\end{bmatrix} \) is the rotation-like matrix and \( \theta = 2\pi k/N \) for \( W_p^k \) or \( \theta = 2\pi r/p \) for \( W_{p^r}^j \). However, \( \cos \theta \) and \( \sin \theta \) in (2-3) are not suitable for directly conversion to the SOPOT representation since the inverse of \( R_\theta \) cannot in general be expressed in terms of SOPOT coefficients. To cope with this, \( R_\theta \) is re-written as follows:
\[
R_\theta = \begin{bmatrix}
  1 -\tan(\theta/2) & 1 \\
  0 & 1 \\
\end{bmatrix} \begin{bmatrix}
  1 & 0 \\
  1 & 1 \\
\end{bmatrix} \begin{bmatrix}
  1 & 1 \\
  \tan(\theta/2) & 0 \\
\end{bmatrix} = R_{\theta'} \quad (2-4)
\]
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 from
\[
R_k = \begin{pmatrix}
1 & -\beta_k \\
0 & 1 \\
\alpha_k & 0 & -1
\end{pmatrix} \Rightarrow R_k^\top,
\]
(2-5)
where $\alpha_k$ and $\beta_k$ are respectively the SOPOT approximations to $\sin \theta$ and $\tan(\theta/2)$ having the form:
\[
z_p = \sum_{i=1}^{r} a_i z_i^p,
\]
(2-6)
where $a_i \in [-1,1], b_i \in [-r,-1,1,...,r]$; $r$ 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 the 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 equal to $\Delta^2/12$, where $\Delta$ is the quantization step-size. In other words, the variance is determined by the number of fractional parts 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:
\[
\sigma^2 = \frac{\Delta^2}{12} = 2^{-(B-1)} / 12.
\]
(3-1)
Without loss of generality, consider the computation of an N-point ($N = p^r$) FFT using the DIT radix-p FFT algorithm as shown in the figure 2. $e^{(m-1)}_j$ for $k = 0,1,...,N/p-1$ and $j = 0,1,...,p-1$ are the signal round-off noises introduced at the k-th output of the j-th (N/p)-point FFT. The superscript (m) indicates the m-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 $X(a) = X^{(m)}(a)$ and $X(a) = X^{(m)}(a)$ for $a = 0,1,...,N-1$. In general, $e^{(m-1)}_j(k)$ are white and identically distributed, by symmetry consideration.

$n^{(m)}(k)$ is the additional signal round-off noise introduced by the multiplication with the “approximated” twiddle factors $W^{(m)}_p$. 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 $W^{(m)}_p$ 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 $W^{(m)}_p$ are $\pm 1$ 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 $n^{(m)}(k)$.

The round-off noise introduced by the rotation-like matrix $R_k$, 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 $\sigma^2_i$ be the variances of the noise sources $r_i$ for $i = 0,1,2$. Assume $\sigma^2_i$ are uncorrelated, white and zero mean, it follows that the output noise variances are given by:
\[
\sigma^2_x = \frac{(1 + \sin^2 \theta \tan^2(\theta/2))\sigma^2_o + \tan^2(\theta/2)\sigma^2_i + \sigma^2_r}{\sigma^2_o + \sigma^2_i + \sigma^2_r},
\]
(3-2)
\[
\sigma^2_r = \frac{(\sin^2 \theta)\sigma^2_o + \sigma^2_i}{\sigma^2_o + \sigma^2_i + \sigma^2_r}.
\]
(3-3)
Thus, the noise variance of $n^{(m)}(k)$ can be written as follows:
\[
VAR[n^{(m)}(k)] = \sigma^2_o + \sigma^2_i.
\]
(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 intermediate 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 $L_q$ bits, then the noise source for $Re(X^{(m)}(q))$ have variance equal to
\[
VAR[Re(X^{(m)}(q))] = \Delta^2/12 = 2^{-L_q}/12.
\]
(3-5a)
Similarly, when the imaginary part of the final result is rounded to $L_q$ bits, then the variance of the noise source for $Im(X^{(m)}(q))$ is given by:
\[
VAR[Im(X^{(m)}(q))] = \Delta^2/12 = 2^{-L_q}/12.
\]
(3-5b)
Hence, we have
\[
VAR[n^{(m)}(k)] = VAR[Re(X^{(m)}(q))] + VAR[Im(X^{(m)}(q))],
\]
(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 $p$ is needed in order to obtain the correct DFTs, but this matter would not be taken into account for our noise model.

$e^{(m-1)}_j$, $n^{(m)}(k)$ and $s^{(m)}(k)$ 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:
\[
VAR[e^{(m)}(q)] = \sum_{j=0}^{\frac{N}{p}} \left[ \sum_{j=0}^{\frac{N}{p}} VAR[e^{(m-1)}_j(k)] \right] \left[ W^{(m)}_p \right]^2 + \sum_{j=0}^{\frac{N}{p}} \left[ \sum_{j=0}^{\frac{N}{p}} VAR[n^{(m)}(k)] \right] \left[ W^{(m)}_p \right]^2 + \sum_{j=0}^{\frac{N}{p}} \left[ \sum_{j=0}^{\frac{N}{p}} VAR(s^{(m)}(q)) \right] \left[ W^{(m)}_p \right]^2.
\]
(3-7)
Further, if we assume $e^{(m-1)}_j(k)$ are identically distributed with variance $VAR[e^{(m-1)}_j(k)]$, then (3-7) simplifies to
\[
VAR[e^{(m)}(q)] = \sum_{j=0}^{\frac{N}{p}} \sum_{j=0}^{\frac{N}{p}} \left[ \sum_{j=0}^{\frac{N}{p}} VAR[n^{(m)}(k)] \right] \left[ W^{(m)}_p \right]^2 + \sum_{j=0}^{\frac{N}{p}} \left[ \sum_{j=0}^{\frac{N}{p}} VAR(s^{(m)}(q)) \right] \left[ W^{(m)}_p \right]^2.
\]
(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:
\[ \hat{A}_j = \left\lfloor \log_{10}(\text{VAR}(e^{j\theta}(q))) + \log_{10}(3/6) \right\rfloor \text{bits.} \quad (3-9) \]

Using (3-8), it is possible to recursively compute the noise power at each output of the different stage, given the values of \( \hat{W}_p^\delta \) and \( \hat{W}_p^{\delta,\epsilon} \), 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 exploited to minimize the hardware complexity \([9,8]\) 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 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 points 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.

**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 \( \{s(n)\}, n = 0,1,...,63 \) 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 0.99998. 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 points 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


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).

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

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).