A COMPUTATIONALLY EFFICIENT HIGH-QUALITY CORDIC BASED DCT

B. Heyne‡‡‡, C. C. Sun∗†, J. Goetze‡‡ and S. J. Ruan∗
‡‡University of Dortmund, Information Processing Lab, Otto-Hahn-Str. 4, 44221 Dortmund, Germany
‡‡§E-Mail: benjamin.heyne@uni-dortmund.de, Phone: +49-231-755-7017, FAX: +49-231-755-7019
∗National Taiwan University of Science and Technology
Department of Electronic Engineering, Taipei 106, Taiwan
†E-Mail: M9302107@mail.ntust.edu.tw, Phone: +886-2-27333141-7204

ABSTRACT
In this paper a computationally efficient and high-quality preserving DCT architecture is presented. It is obtained by optimizing the Loeffler DCT based on the Cordic algorithm. The computational complexity is reduced from 11 multiply and 29 add operations (Loeffler DCT) to 38 add and 16 shift operations (which is similar to the complexity of the binDCT). The experimental results show that the proposed DCT algorithm not only reduces the computational complexity significantly, but also retains the good transformation quality of the Loeffler DCT. Therefore, the proposed Cordic based Loeffler DCT can be used in low-power and high-quality CODECs, especially in battery-based systems.

1. INTRODUCTION
Recently, many kinds of digital image processing and video compression techniques have been presented in the literature, such as JPEG, Digital Watermark, MPEG and H.263 [1, 2]. All the above standards require the Discrete Cosine Transform (DCT) [1] to aid image/video compression. Therefore, the DCT has become more and more important in today’s image/video processing designs.

In the past few years, much research has been done on low power DCT designs [3–8]. One of the most popular ways to realize the fast DCT (FDCT) is to use the Flow-Graph Algorithm (FGA) for VLSI-implementation [9, 10]. The Loeffler 1-D 8-point DCT algorithm based on FGA that requires only 11 multiply and 29 add operations. However, the multiplications consume about 40% of the power and almost 45% of the total area [12]. In this regard, Tran [13] proposed the binDCT which approximates multiplications with add and shift operations. It only consumes about 38% of the power of the Loeffler DCT. However, it also looses about 3 dB in PSNR compared to the Loeffler DCT [12].

Jeong et al. [6] have proposed a Cordic based implementation of the DCT. CoOrdinate Rotation Digital Computer (Cordic) is an algorithm that can be used for the evaluation of various functions in signal processing [14, 15]. In addition, the Cordic algorithm is highly suited for VLSI-implementation.

In this paper we propose a computationally efficient and high-quality Cordic based Loeffler DCT architecture, which is optimized by taking advantage of certain properties of the Cordic algorithm and its implementations [16]. It only requires 38 add and 16 shift operations. The resulting DCT algorithm not only reduces computational complexity significantly, but also retains the good transformation quality of the Loeffler DCT. Therefore, the presented Cordic based Loeffler DCT implementation is especially suited for low-power and high-quality CODECs.

This paper is organized as follows. Section 2 briefly introduces the algorithms of the DCT, Loeffler DCT and Cordic based DCT. In section 3, we will present the proposed Cordic based Loeffler DCT algorithm. The experimental results are shown in Section 4, while Section 5 concludes this paper.

2. DCT ALGORITHMS

2.1 The DCT Background
The two dimensional DCT in Eq.(1) transforms an 8 × 8 block sample from spatial domain f(x,y) into frequency domain F(k,l).

\[ F(k,l) = \frac{1}{4} C(k) C(l) \sum_{x=0}^{7} \sum_{y=0}^{7} f(x,y) \cos \left( \frac{(2x+1)\pi}{16} \right) \cos \left( \frac{(2y+1)\pi}{16} \right) \] (1)

\[ C(m) = \begin{cases} \frac{1}{2} & \text{if } m = 0 \\ 1 & \text{otherwise.} \end{cases} \]

Since computing the above 2-D DCT by using matrix multiplication requires \(8^4\) multiplications, a commonly used approach in hardware designs to reduce the computational complexity is row-column decomposition. The decomposition performs row-wise one-dimensional (1-D) transform followed by column-wise 1-D transform with intermediate transposition. An 8-point 1-D DCT can be expressed as follows:

\[ F(k) = \frac{1}{2} C(k) \sum_{x=0}^{7} f(x) \cos \left( \frac{(2x+1)\pi}{16} \right) \] (2)

\[ C(k) = \begin{cases} \frac{1}{2} & \text{if } k = 0 \\ 1 & \text{otherwise.} \end{cases} \]

This decomposition approach has two advantages. Firstly the number of operations is significantly reduced. Secondly, with regard to the implementation, the original 1-D DCT can be replaced easier by more efficient DCT algorithms.

2.2 MAC based Loeffler DCT
Many 1-D flow graph algorithms have been reported in the literature [9, 10]. The Loeffler 1-D 8-point DCT algorithm

\[ F(k,l) = \frac{1}{4} C(k) C(l) \sum_{x=0}^{7} \sum_{y=0}^{7} f(x,y) \cos \left( \frac{(2x+1)\pi}{16} \right) \cos \left( \frac{(2y+1)\pi}{16} \right) \] (1)

\[ C(m) = \begin{cases} \frac{1}{2} & \text{if } m = 0 \\ 1 & \text{otherwise.} \end{cases} \]

Since computing the above 2-D DCT by using matrix multiplication requires \(8^4\) multiplications, a commonly used approach in hardware designs to reduce the computational complexity is row-column decomposition. The decomposition performs row-wise one-dimensional (1-D) transform followed by column-wise 1-D transform with intermediate transposition. An 8-point 1-D DCT can be expressed as follows:

\[ F(k) = \frac{1}{2} C(k) \sum_{x=0}^{7} f(x) \cos \left( \frac{(2x+1)\pi}{16} \right) \] (2)

\[ C(k) = \begin{cases} \frac{1}{2} & \text{if } k = 0 \\ 1 & \text{otherwise.} \end{cases} \]

This decomposition approach has two advantages. Firstly the number of operations is significantly reduced. Secondly, with regard to the implementation, the original 1-D DCT can be replaced easier by more efficient DCT algorithms.

2.2 MAC based Loeffler DCT
Many 1-D flow graph algorithms have been reported in the literature [9, 10]. The Loeffler 1-D 8-point DCT algorithm

\[ F(k,l) = \frac{1}{4} C(k) C(l) \sum_{x=0}^{7} \sum_{y=0}^{7} f(x,y) \cos \left( \frac{(2x+1)\pi}{16} \right) \cos \left( \frac{(2y+1)\pi}{16} \right) \] (1)

\[ C(m) = \begin{cases} \frac{1}{2} & \text{if } m = 0 \\ 1 & \text{otherwise.} \end{cases} \]

Since computing the above 2-D DCT by using matrix multiplication requires \(8^4\) multiplications, a commonly used approach in hardware designs to reduce the computational complexity is row-column decomposition. The decomposition performs row-wise one-dimensional (1-D) transform followed by column-wise 1-D transform with intermediate transposition. An 8-point 1-D DCT can be expressed as follows:

\[ F(k) = \frac{1}{2} C(k) \sum_{x=0}^{7} f(x) \cos \left( \frac{(2x+1)\pi}{16} \right) \] (2)

\[ C(k) = \begin{cases} \frac{1}{2} & \text{if } k = 0 \\ 1 & \text{otherwise.} \end{cases} \]

This decomposition approach has two advantages. Firstly the number of operations is significantly reduced. Secondly, with regard to the implementation, the original 1-D DCT can be replaced easier by more efficient DCT algorithms.
The vector rotation can be performed using the iterative algorithm, that is rotating a vector \((x_i, y_i)\) by an angle \(\theta\), the circular rotation angle is described as

\[
\theta = \sum_i \sigma_i \cdot \tan^{-1}(2^{-i})
\]

with \(\sigma_i = 1, -1\).

Then, the vector rotation can be performed using the iterative equation given in [6, 18]:

\[
\begin{align*}
x_{i+1} &= x_i - \sigma_i \cdot y_i \cdot 2^{-i} \\
y_{i+1} &= y_i + \sigma_i \cdot x_i \cdot 2^{-i}.
\end{align*}
\]

In equation 4, only shift and add operations are required in digital hardware. Next, the results of the rotation iterations need to be compensated (scaled) by a compensation factor \(s\). This is also done using an iterative approach:

\[
x_{i+1} = x_i (1 + y_i \cdot F_i) \\
y_{i+1} = y_i (1 + y_i \cdot F_i)
\]

with \(\prod_i (1 + y_i \cdot F_i) \approx s\) and \(y_i = (0, 1, -1), F_i = 2^{-i}\).

When using the Cordic to replace the multiplications of the 8-point DCT the angles \(\theta_i\) are fixed. Therefore, we can skip some unnecessary iterations without losing accuracy. Table 1 shows the detailed number of iterations and compensations for the Cordic based algorithm [6]. Although the Cordic based DCT can reduce the number of computations in image/video compression, it still needs more operations than the binDCT does [13].

### 3. CORDIC BASED LOEFFLER DCT

Based on our previous work about Cordic based FFTs [19, 20], we now propose an optimized Cordic based Loeffler DCT in this paper. This implementation requires only 38 add and 16 shift operations. We have taken the original Loeffler DCT as the starting point for our optimization, because the theoretical lower bound of the number of multiplications required for the 1-D 8-point DCT had been proven to be 11 [21].

In order to derive the proposed algorithm, we first consider the butterfly at the beginning of Loeffler’s flow graph as shown in Figure 1. In this case the butterfly can be expressed as:

![Figure 1: Flow graph of an 8-point Loeffler DCT.](image-url)
At first, due to the special structure of the Loeffler DCT, the scaling of $\sqrt{2}$ and the five needed compensation steps as shown in Table 1 can be performed at the end of the flow graph for the angle $\theta = \pi/4$. In other words, the $\pi/4$ rotation only needs two add operations to carry out the Cordic rotation.

Secondly, for the angle $\theta = 3\pi/8$ we reduce the number of iterations to three and also shift all compensation steps to the quantizer. Although the optimized $3\pi/8$ rotation will decrease the quality of the results, the influences are not noticeable in video sequence streams or image compression.

Thirdly, when we take a close look at the angle $\theta = \pi/16$, it can be easily observed that the needed compensation of the $\pi/16$ rotation is very close to one. Thus, we can ignore the compensation steps of the $\pi/16$ rotation. Therefore, it only needs two iterations in the Cordic calculation. Unfortunately we can not shift any compensation steps of the $3\pi/16 \leftrightarrow 3\pi/4 \leftrightarrow 3\pi/8$ rotation to the end of the graph, due to the data correlation between the following stages of the $\pi/16$ and $3\pi/16$ rotations. However, we still can ignore some unnoticeable iterations and compensation steps to reduce the computational complexity of the angle $\theta = 3\pi/16$.

Table 2 shows the summary of Cordic iterations and compensation steps for the proposed Cordic based Loeffler DCT. In Fig. 5 the optimized flow graph is shown, including the scaling factors incorporated into the quantization table. It only requires 38 add and 16 shift operations to realize the DCT transformation. In short, we try to ignore some
unnoticeable iterations and shift the compensation steps of each angle to the quantizer to derive the optimized Cordic based Loeffler DCT. Moreover, the proposed DCT algorithm not only reduces the computational complexity significantly, but also keeps the high transformation quality as well as the original Loeffler DCT does.

Therefore, the proposed DCT algorithm has the same computational complexity as the binDCT, but as shown in the next Section it can perform as well as the Loeffler DCT in quality.

4. EXPERIMENTAL RESULTS

In our experiments we have used different criteria to evaluate four architectures: Loeffler DCT, Cordic based DCT, binDCT-C5 and Cordic based Loeffler DCT. Table 3 summarizes the number of operations of each DCT architecture. It can be easily observed from Table 3 that the proposed DCT reduces the computational complexity significantly compared to the original Loeffler DCT and Cordic based DCT.

<table>
<thead>
<tr>
<th>DCT type</th>
<th>Operation</th>
<th>Mult</th>
<th>Add</th>
<th>Shift</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loeffler</td>
<td>11</td>
<td>29</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Cordic [6]</td>
<td>0</td>
<td>104</td>
<td>82</td>
<td></td>
</tr>
<tr>
<td>Cordic Loeffler</td>
<td>0</td>
<td>38</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>binDCT-C5 [22]</td>
<td>0</td>
<td>36</td>
<td>17</td>
<td></td>
</tr>
</tbody>
</table>

In order to demonstrate the quality features of the proposed DCT algorithm, we have also applied it to the video coding standard MPEG 4, by using a publicly available XVID CODEC software [23]. The DCT in the CODEC of the selected XVID implementation is based on Loeffler’s factorization with floating-point multiplications. In this part we have applied each DCT algorithm to the XVID software, and simulated with some well-known video sequences to show the ability of the proposed approach. Figure 6 shows the average PSNR results of four DCT algorithms from low to high compression ratio (i.e. quantization steps from 1 to 10) with the “Foreman” video sequence.

It can be seen from the PSNR simulation results, that the proposed algorithm performs as well as the Loeffler DCT does. Also, the average PSNR is about 2 dB higher than that of the binDCT-C5. In summary, the overall performance of the Cordic based Loeffler DCT is very similar to the Loeffler DCT in terms of video quality. However, it only requires the same computational complexity as the binDCT-C5 does.

In order to analyze the performance of the proposed Cordic based Loeffler DCT, we have modeled the four different DCT architectures as RTL. After synthesizing with Synopsys Design Compiler, we have used Synopsys PrimePower to estimate the power consumption at gate-level. The simulation results are shown in Table 4. Some important points can be observed easily. Firstly, the proposed DCT architecture only consumed 19% of the area and about 16% of the power of the original Loeffler DCT. Secondly, the proposed DCT architecture occupied the same area as the binDCT-C5. However, it has only about 59% of the power dissipation and half the delay time of the binDCT-C5.

Finally, it is worth to note that the proposed DCT not only reduces the computational complexity significantly, but also achieves the best performance in all criteria.

5. CONCLUSION

In this paper a low complexity and high quality DCT transformation based on the Cordic algorithm is presented. The proposed Cordic based Loefffer DCT architecture only requires 38 add and 16 shift operations to carry out the DCT transformation, which is about the same complexity as the binDCT-C5’s. The proposed algorithm not only reduces the computational complexity significantly compared to the original Loeffler DCT, it also keeps the good quality transformation result. In this regard, the proposed DCT algorithm is...
Table 4: Power, Area and Time delay simulation results at gate-level.

<table>
<thead>
<tr>
<th>Measures</th>
<th>DCT Arch.</th>
<th>Loeffler</th>
<th>Cordic based [6]</th>
<th>Cordic based Loeffler</th>
<th>binDCT-C5 [22]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power (mW)</td>
<td></td>
<td>3.557</td>
<td>1.954</td>
<td>0.5616</td>
<td>0.9604</td>
</tr>
<tr>
<td>Area (GateCount)</td>
<td></td>
<td>15.06K</td>
<td>6.66K</td>
<td>2.81K</td>
<td>2.83K</td>
</tr>
<tr>
<td>Delay (ns)</td>
<td></td>
<td>13.49</td>
<td>15.08</td>
<td>8.37</td>
<td>12.17</td>
</tr>
</tbody>
</table>

TSMC 0.13-μm at 1.2V without pipelining.

very suitable for low-power and high quality CODECs.

REFERENCES