An Efficient Implementation of the 1D DCT using FPGA Technology

Hassan EL-Banna*  Alaa A. EL-Fattah*  Waleed Fakhr**

*Electronics Research Institute, Cairo, Egypt  ** Arab Academy for Science and Technology, Cairo, Egypt

ABSTRACT

This paper describes and represents different algorithms and efficient implementation of One Dimensional 8-point Discrete Cosine Transform on Field Programmable Gate Arrays. One of the main objectives is to minimize the complexity of operations as much as possible while maintaining low delays and high speed throughput. Distributed Arithmetic is a powerful technique that has been used for fast and efficient implementation of 1D DCT on FPGA.

1. INTRODUCTION

The discrete cosine transform DCT forms a key role in several image compression standards including JPEG [1] for still picture compression, ITU H.261 [2] and H.263 for teleconferencing, and ISO MPEG-1 and MPEG-2 [3] for audio, visual compression and communication. Some speech enhancement techniques use DCT [4]. In addition to that, 1D DCT has most often been used in 2D DCT, by employing the row-column decomposition, exploiting the fact that the formula of the 2D DCT is separable, which means that it can be broken into two sequential 1D DCT operations, one along the row vector and the second along the column vector of the preceding row vector results. The Row-Column decomposition method is the most common method deployed for computing the 2D DCT, and implementations usually focus on optimizing the 1D DCT so that the Row-Column 2D DCT implementation performs better when using the optimized 1D DCT block along rows and columns.

This paper is organized as follows. Section 2 provides a review of 1D DCT computation. Section 3 reviews some DCT algorithms. Section 4 explains the Distributed Arithmetic technique. The last section shows the 1D DCT Architecture we implemented using the Distributed Arithmetic.

2. The 1D Discrete Cosine Transform

The Discrete Cosine Transform has long been the basic transform coding method for the JPEG and MPEG standards. It helps separate the image into parts (or spectral sub-bands) of differing importance - with respect to the spatial quality of the image. In that respect, it is similar to the Discrete Fourier Transform since it transforms a signal or image from the spatial domain to the frequency domain. However one primary advantage of the DCT over the DFT is that the former involves only real multiplications, which reduces the total number of required multiplications, unlike the latter. Another advantage lies in the fact that for most images much of the signal energy lies at low frequencies, and are often small - small enough to be neglected with little visible distortion. The DCT does a better job of concentrating energy into lower order coefficients than does the DFT for image data. This characteristic of the DCT, referred to as energy compaction efficiency, along with other advantages resulted in the JPEG and MPEG standards adopting the DCT as a standard for image compression.

The N-point 1-D DCT is defined by [5]:

\[ Y(k) = \frac{2}{N} \sum_{n=0}^{N-1} y(n) \cos \left( \frac{(2n+1)k\pi}{2N} \right) \quad k=0,1,..,N-1 \] (1)

where

\[ C_k = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } k = 0 \\ 1 & \text{for } k \neq 0 \end{cases} \]

Real-time implementation of the DCT operation is highly computationally intensive. Accordingly, much effort has been directed to the development of suitable cost effective VLSI architectures to perform this. Traditionally the focus has been on reducing the number of multiplications required. Additional design criteria has included minimizing the complexity of control logic, memory requirements, power consumption and complexity of interconnect.

3. Some DCT algorithms

3.1 Chen et al Algorithm

The 8-point DCT can be written as a matrix transform.

\[ Y = AX \]

Where

\[ A = \begin{bmatrix} d & d & d & d & d & d & d & d \\ a & c & e & g & -g & -c & -e & -a \\ b & f & -f & -b & -g & -c & e & a \\ c & -g & -a & -e & c & a & -g & -c \\ d & -d & -d & -d & -d & -d & -d & -d \\ e & -g & -c & -e & c & a & -g & -c \\ f & -b & -f & -f & -b & -g & -a & -c \\ g & -c & e & -a & a & -e & c & -g \end{bmatrix} \]
the multiplier coefficients are given by:

<table>
<thead>
<tr>
<th>a</th>
<th>c₁</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>c₂</td>
<td>p</td>
</tr>
<tr>
<td>c</td>
<td>c₃</td>
<td>q</td>
</tr>
<tr>
<td>d</td>
<td>c₄</td>
<td>r</td>
</tr>
<tr>
<td>e</td>
<td>c₅</td>
<td>s</td>
</tr>
<tr>
<td>f</td>
<td>c₆</td>
<td>t</td>
</tr>
<tr>
<td>g</td>
<td>c₇</td>
<td>u</td>
</tr>
<tr>
<td>h</td>
<td>c₈</td>
<td>v</td>
</tr>
<tr>
<td>i</td>
<td>c₉</td>
<td>w</td>
</tr>
<tr>
<td>j</td>
<td>1/2c₁</td>
<td>x</td>
</tr>
<tr>
<td>k</td>
<td>1/2c₂</td>
<td>y</td>
</tr>
<tr>
<td>m</td>
<td>1/2c₆</td>
<td>z</td>
</tr>
</tbody>
</table>

where

\[ C_n = \cos\left(\frac{n\pi}{16}\right) \]

Due to the Symmetry of the (8 X 8) multiplication matrix, it can be replaced by two (4x4) x (4x4) matrices which can be computed in parallel, as can the sums and differences forming the vectors below:

\[
\begin{bmatrix}
y_0 \\
y_1 \\
y_2 \\
y_3 \\
y_4 \\
y_5 \\
y_6 \\
y_7 \\
\end{bmatrix} =
\begin{bmatrix}
d & d & d & d \\
b & f & -f & -b \\
d & -d & -d & d \\
f & -b & b & -f \\
a & c & e & g \\
e & -a & g & e \\
g & -c & e & -a \\
\end{bmatrix}
\begin{bmatrix}
x_0 + x_1 \\
x_1 + x_0 \\
x_3 + x_5 \\
x_5 + x_3 \\
x_0 - x_7 \\
x_1 - x_6 \\
x_2 - x_5 \\
x_3 - x_4 \\
\end{bmatrix}
\]

The implementations by Madisetti and Willson [6], Uramoto et al [7], Matsui et al [8], and Jang et al [9] are based upon this decomposition which requires 32 multiplications. However, Madisetti and Willson reduce the number of multiplications to 28.

The frequently referenced algorithm by Chen et al [10] only requires 16 multiplications with 2 multiplications on the critical path. The data-flow graph for Chen's algorithm is shown in Figure 1.

3.2 Lee Algorithm

Lee algorithm [11] is based on the matrix representation. In fact, the first step is nothing than a butterfly decomposition yielding to an even and an odd part. The even part will be just a 1-D DCT of order N/2. While, the odd part will be computed through a matrix multiplication.

Figure 2 illustrates 1-D DCT of order 8.

For 1-D DCT of order N=8, the number of operation necessarily for these algorithm will be 32 multiplications and 32 additions.

3.3 Loeffler Algorithm

Based on equation (1), Loeffler [12] has proposed a new class of a fast 1D-DCT algorithm that requires 11 multiplications and 29 additions only.

An algorithm of this class is shown in Figure 3.

Figure 3 illustrates 1-D DCT of order 8.

The stages of the algorithm numbered 0 to 3 are parts that have to be executed in serial mode due to the data dependency. However, computation within the first stage can be paralleled. In stage1, the algorithm splits in two parts. One for the even coefficients, the other for the odd ones. The even part is nothing else than a 4 points DCT, again separated in even and odd parts in stage2.

The second building block can be calculated using only 3 multiplications and 2 additions only instead of 4 multiplications and 2 additions. This can be done by using the equivalence showed in the following equations:

\[
y_o = a.x_o + b.x_1 = (b - a).x_o + a.(x_o + x_1)
\]

\[
y_1 = -b.x_o + a.x_1 = -(b + a).x_o + a.(x_o + x_1)
\]
The constant $C$ was chosen to be equal to $\sqrt{2}$ which allows the first DCT coefficient to be evaluated without any multiplication. Figure 4 explains the building blocks of the algorithm.

![Figure 4: Algorithm operators](image1)

4. Distributed Arithmetic

Distributed Arithmetic is a very commonly used technique where Multiply-Accumulate plays predominant role in the operation, especially true with signal processing applications. Typically it serves to eliminate multiplications and replace them with adds, which is useful since a multiplication consumes much more time than an add.

An example of the result is a case where $N$ multiplies followed by an $N$-input add has been replaced by a series of $N$-input adds followed by a single multiply.

5. The Architecture Implemented

From Chen et al Algorithm, we can find that the transform matrix $A$ could be divided to 2 smaller matrices. By using the Distributed Algorithm technique these 2 matrices could be more simplified to give the following equations:

\[
y_l = \sum_{k=0}^{3} A_{k,l}(x_k + x_{7-k}) \quad \text{for } l \text{ even}
\]

\[
y_l = \sum_{k=0}^{3} A_{k,l}(x_k - x_{7-k}) \quad \text{for } l \text{ odd}
\]

The Architecture of the 1D DCT is shown in Figure 5.

![Figure 5: 8points 1D DCT Architecture](image2)

The 4-Product MAC could be designed using the conventional arithmetic as shown in Figure 6, or could be designed by using the serial distributed arithmetic as shown in Figure 7.

![Figure 6: 4-product MAC using conventional arithmetic](image3)

![Figure 7: 4-product MAC using Serial Distributed Arithmetic](image4)
The following flow chart briefly explains the hardware implementation of the 1D DCT that uses Arithmetic Distribution we have done.

![Flow Chart]

we have 8 inputs, each input is 8bit width. First the inputs are registered. Then they are added and subtracted according to the matrix of Chen et al. The Bit Serial Architecture is primarily used in the context of multipliers, these are architectures where a single bit bit of each input word is transmitted during each processing cycle. This reduces I/O, however an n-bit word requires n-processing cycles for transmission. The input word to the bit serial architecture is 10 bit. This means we need 10 cycles. Lookup tables (ROMS) contain partial product terms that are indexed using the bit-serial input from the multiplier. An accumulator is used to add each partial product term. The VHDL code was written using FPGA Advantage and was implemented on Xilinx SpartanII FPGA, which uses look-up tables and therefore should make an efficient use of the design.

References