LOW POWER DCT ARCHITECTURE FOR IMAGE/VIDEO CODERS

Vikram Angala¹, Harika.T²

¹Assistant Professor, Department of E.C.E, SCITS, Karimnagar, India.
²Assistant Professor NIGAMA Engineering College, Karimnagar, India.

ABSTRACT

As 2-D DCT algorithms are the most typical for multimedia applications, the main focus of this project will be on the efficient hardware implementations of 2-D DCT. As the number of applications that require higher dimensional DCT algorithms is growing, a special attention will be given to the algorithms that are easily extensible to higher dimensional cases. The goal of this project will be to implement DCT on FPGA. In this thesis a 2-D DCT core is considered and it is two dimensional discrete cosine transform implementation designed for use in compression systems like JPEG. Core has simple input interface. Design is synchronous, with single positive clock edge and no internal tri-state buffers. Latency between first latched input data and first DCT transformed output is 85 clock cycles. Design is internally pipelined, when the pipeline is full 64 point input data is transformed in 64 clock cycles to 2D DCT values. Core uses double buffered (ping-pong scheme) RAM for storing intermediate product results after first DCT stage for maximized performance. This way both 1D DCT units can work in parallel effectively creating dual stage global pipeline. 2D-DCT core takes 8 bit input data and produces 12 bit output using 12 bit DCT matrix coefficients. The aim of this project is to create an architecture and code in Verilog, and implement the core in Xilinx Virtex4 FPGA.

Keywords: DCT – Discrete Cosine Transform, JPEG – Joint Photographic Expert Group, FPGA – Field Programmable Gate Array.

1. INTRODUCTION

The DCT is a mathematical operation that transform a set of data, which is sampled at a given sampling rate, to it's frequency components. The number of samples should be finite, and power of two for optimal computation time. A discrete cosine transform (DCT) is a Fourier-related transform similar to the discrete Fourier transform (DFT), but using only real numbers. DCTs are equivalent to DFTs of roughly twice the length, operating on real data with even symmetry (since the Fourier transform of a real and even function is real and even), where in some variants the input and/or output data are shifted by half a sample. There are eight standard DCT variants, of which four are common. The most common variant of discrete cosine transform is the type-II DCT, which is often called simply "the DCT"; its inverse, the type-III DCT, is correspondingly often called simply "the inverse DCT" or "the IDCT". Although the direct application of these formulas would require (N²) operations, it is possible to compute the same thing with only (N log N) complexity by factoring the computation similar to the fast Fourier transform (FFT). One can also compute DCTs via FFTs combined with (N) pre- and post-processing steps. Discrete cosine transform (DCT) is widely used transform in image processing, especially for compression. Some of the applications of DCT involve still image compression and compression of individual video frames, while multidimensional DCT is mostly used for compression of video streams and volume spaces. Transform is also useful for transferring multidimensional data to DCT frequency domain, where different operations, like spread-spectrum data watermarking, can be performed in easier and more efficient manner. The technology is used in several known video codec’s such as Motion JPEG, MPEG-1, MPEG2, MPEG-4, H.261 and H.263. DCT is a compression algorithm that tests an image at regular intervals, by converting spatial amplitude into spatial frequency data. Lower frequencies contributes with more picture information than higher frequencies does. This results in the possibility to change an image to frequent components and throw away a lot of high frequencies. By this we can reduce the number of data that are needed to describe the image without losing too much quality. Hardware implementations are especially interesting for the realization of highly parallel algorithms that can achieve much higher throughput than software solutions. In addition, special purpose DCT hardware discharges the computational load from the processor and therefore improves the performance of complete multimedia system. The throughput is directly influencing the quality of experience of multimedia content. As 2-D DCT algorithms are the most typical for multimedia applications, the main focus of this project will be on the efficient hardware implementations of 2-D DCT.
As the number of applications that require higher dimensional DCT algorithms is growing, a special attention will be given to the algorithms that are easily extensible to higher dimensional cases.

1.1 Comparison between DFT and DCT

In DFT (figure b) periodic sequence is not smooth. There usually exist discontinuities at the beginning and end of each period. These end head discontinuities cause a high frequency distribution in the corresponding DFT. On the contrary, in the DCT (figure d) periodic sequence does not have this type of discontinuity due to flipping over the given finite sequence. As a result, there is no high-frequency component corresponding to the end-head finite discontinuities. Hence, the DCT possesses better energy compaction in the low frequencies than the DFT. By better energy compaction, we mean more energy compacted in a fraction of transform coefficients. For instance, it is known that the most energy of an image is contained in small region of low frequency in DFT domain.

1.2 Applications

DCT is used in large number of image and signal processing applications including
1. JPEG standard
2. MPEG standards
3. H.261 and H.263 video conferencing standards
4. DVD, VCD, SVCD, HDTV etc.

The DCT, and in particular the DCT-II, is often used in signal and image processing, especially for lossy data compression, because it has a strong "energy compaction" property: most of the signal information tends to be concentrated in a few low-frequency components of the DCT, approaching the optimal Karhunen-Loève transform for signals based on certain limits of Markov processes. A related transform, the modified discrete cosine transform (MDCT), is used in AAC, Vorbis, and MP3 audio compression. DCTs are also widely employed in solving partial differential equations by spectral methods, where the different variants of the DCT correspond to slightly different even/odd boundary conditions at the two ends of the array.

2. DISCRETE COSINE TRANSFORM

2.1 Discrete Cosine Transform

The 2N-DFT of \( x'(n) \) is

\[
y(k) = \frac{1}{\sqrt{2N}} \sum_{N=0}^{2N-1} X^1(n)e^{-j(2\pi/2N)N} \sum_{n=0}^{N-1} x(n) \cos \left( \frac{(2n+1)k\pi}{2N} \right)
\]

By comparing it with \( y(k) \), it can be seen that

\[
y(k) = \alpha(k) e^{-j k\pi/2N} y'(k)
\]
Conceptually, a one-dimensional DCT can be thought of as taking the Fourier transform and retaining only the real (cosine) part. The two-dimensional DCT can be obtained by performing a one-dimensional DCT on the columns and then, a one-dimensional DCT on the rows. The transformed output from the two-dimensional DCT is ordered so that the mean value (the DC coefficient) is in the upper left corner of the 8 x 8 coefficient block, and the higher frequency coefficients progress by distance from the DC coefficient. Higher row numbers represents higher vertical frequencies, and higher column numbers represents higher horizontal frequencies.

### 2.1.1 One-Dimensional DCT

DCT of One-Dimensional function of N elements is

\[
U(K) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} u(n) \cos \left( \frac{(2n+1)K\pi}{2N} \right)
\]

In case that \(K\neq 0\), we get

\[
U(K) = \frac{2}{\sqrt{N}} \sum_{n=0}^{N-1} u(n) \cos \left( \frac{(2n+1)K\pi}{2N} \right)
\]

The 2N-DFT of \(x'(n)\) is

\[
y(k) = \frac{1}{2\sqrt{N}} \sum_{n=0}^{2N-1} x'(n)e^{-j(2\pi kn/2N)} = \frac{1}{2\sqrt{N}} e^{jk\pi/2N} \sum_{n=0}^{N-1} x(n) \cos \left( \frac{(2N+1)k\pi}{2N} \right)
\]

By comparing it with \(y(k)\), it can be seen that

### 2.1.2 Two-Dimensional DCT

In two dimensions, the DCT can be expressed in the form

\[
y(k,l) = \frac{2}{N} \alpha(k)\alpha(l) \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} x(m,n) \cos \frac{\pi k(2m+1)}{2N} \cos \frac{\pi l(2n+1)}{2N}
\]

where \(x(m,n)\) is a \(N \times N\) field, \(k, l, m, n\) all range from 0 to \(N-1\). Note that the transformation kernels are separable, so that the 2-D DCT can be conveniently performed in two steps, each of which involves a 1-D DCT. As in the one-dimensional case DCT coding with an even symmetry end extension has fewer edge-effect problems than DFT image coding. Since the 2D DCT can be computed by applying 1D transforms separately to the rows and columns, we say that the 2D DCT is separable in the two dimensions. As in the 1D case, each element \(F(u, v)\) of the transform is the inner product of the input and a basis function, but in this case, the basic functions are \(M \times N\) matrices. Each 2D-basis matrix is the outer product of two of the 1D basis vectors.
2.2 Computing the 2D DCT

- Factoring reduces problem to a series of 1D DCTs
- Apply 1D DCT (Vertically) to Columns
- Apply 1D DCT (Horizontally) to resultant Vertical DCT above.
- Or alternatively horizontal to vertical.

The equations are given by:

\[
C_{x}(k_1,k_2) = 4 \frac{e^{j\pi k_1}}{N_1} \frac{e^{j\pi k_2}}{N_2} \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} X(n_1,n_2) \cos \left( \frac{\pi}{2N_2} (2n_2 + 1) \right) \cos \left( \frac{\pi}{2N_1} (2n_1 + 1) \right)
\]

For 0 ≤ k1 ≤ N1-1;
0 ≤ k2 ≤ N2-1

Otherwise \( C_x(k_1,k_2) = 0. \)

\( \mathcal{E} = \left( \frac{1}{\sqrt{2}} \right) \) for \( k = 0. \)

= 1 otherwise

Where:

- \( N1 \) is the number of rows in the input matrix.
- \( N2 \) is the number of columns in the input matrix.
• n1 is the row index in the input matrix.
• n2 is the column index in the input matrix.
• k1 is the row index in the output cosine matrix.
• k2 is the column index in the output cosine matrix.
• X (n1, n2) is the time domain data.
• \( Cx \) (k1, k2) is the transform domain coefficient.

From the above equation, it is obvious that to calculate one transform coefficient we need (N1. N2) additions and (N1. N2) multiplication. There are (k1. k2) transform coefficients, which totally needs (N1. N2. k1. k2) additions and (N1. N2. k1. k2) multiplication. For the case where the output transform coefficients matrix has the same dimensions as the input data matrix, we will have k1 = N1 and k2 = N2. Thus to calculate all the transform domain coefficients, we need (N1N2)^2 additions and (N1N2)^2 multiplication, which is a large number of calculations leading to big delay.

3. ARCHITECTURE
3.1 Architecture

The 2D-DCT core is two dimensional discrete cosine transform implementation designed for use in compression systems like JPEG. Architecture is based on parallel distributed arithmetic with butterfly computation. Done as row/column decomposition where two 1D DCT units are connected through transposition matrix memory. Core has simple input interface. Design is synchronous, with single positive clock edge and no internal tri-state buffers. Latency between first latched input data and first DCT transformed output is 85 clock cycles. Design is internally pipelined, when the pipeline is full 64 point input data is transformed in 64 clock cycles to 2D DCT values. Core uses double buffered (ping-pong scheme) RAM for storing intermediate product results after first DCT stage for maximized performance. This way both 1D DCT unit can work in parallel effectively creating dual stage global pipeline. MDCT core takes 8 bit input data and produces 12 bit output using 12 bit DCT matrix coefficients. This may be enhanced to be configurable in the future. Self checking test bench is written for testing operation of the 2D-DCT core. Test bench takes matlab-converted bitmap image; 2D-DCT core DCT-transforms it and test bench compares input image to reconstructed one by performing behavioural IDCT. Peak signal to noise ratio is computed to see how big error is introduced by fixed point arithmetic used in core. Test bench created DCT images can be converted to JPEG format by Matlab scripts.

This section describes the architecture of the 2D-DCT. Below is I/O schematic of 2D-DCT core.

3.1.1 Block diagram

Block diagram is presented below describing the top level of the design.
1D-DCT block
This block performs one dimensional DCT on the data. Two blocks of 1D-DCT are used to perform 2D-DCT. 1<sup>st</sup> DCT block performs one dimensional DCT on input data; stores the results in transpose RAMs. 2<sup>nd</sup> DCT block performs another level of DCT on the data stored in the transpose RAM.

**Transpose RAM block**
Transposition RAM is double buffered, that is, when 2<sup>nd</sup> stage of DCT reads out data from transposition memory 1, 1<sup>st</sup> DCT stage can populate 2<sup>nd</sup> transposition memory with new data.

**ROM Blocks**
ROM blocks are used to store pre-computed MAC results and grab them as needed by the DCT stages.

**DBUFCTL block**
DBUFCTL block is essentially a memory arbiter between 1D DCT stages. Each stage can request memory buffer for read/write and basing on availability one of two RAM buffers is granted. 2D-DCT core architecture is based on two 1D DCT units connected through transpose matrix RAM. Transposition RAM is double buffered, that is, when 2<sup>nd</sup> stage of DCT reads out data from transposition memory 1, 1<sup>st</sup> DCT stage can populate 2<sup>nd</sup> transposition memory with new data. This enables creation of dual stage global pipeline where every stage consists of 1D DCT and transposition memory. 1D DCT units are not internally pipelined; they use parallel distributed arithmetic with butterfly computation to compute DCT values. Distributed arithmetic (DA) is a bit level rearrangement of a multiply accumulate to hide the multiplications. It is a powerful technique for reducing the size of a parallel hardware multiply-accumulate that is well suited to FPGA designs. Because of parallel DA they need considerable amount of ROM memories to compute one DCT value in single clock cycle. Design based on distributed arithmetic does not use any multipliers for computing MAC (multiply and accumulate), instead it stores pre-computed MAC results in ROM memory and grab them as needed. DBUFCTL block is essentially a memory arbiter between 1D DCT stages. Each stage can request memory buffer for read/write and basing on availability one of two RAM buffers is granted. Every RAM has 10 bit width data and 64 memory cells. Design also uses ROM memories for storing pre-computed MAC coefficients. ROMs can be either asynchronous or synchronous. ROME and ROMO memory models included in core are synchronous synthesizable ROM memory models which can be used for simulation and implementation. To have asynchronous distributed ROMs clocking and input address register from these ROMs can be removed.

### 3.2 PROPOSED DCT ARCHITECTURE

The proposed DCT architecture targets power efficiency by minimizing the number of arithmetic operations as well as the bit-width for the arithmetic logic. For 2-D DCT, the row column decomposition technique can be used [1]. It reduces the complexity of DCT by a factor of four. Our 1-D DCT design that minimizes the number of arithmetic operations is presented in the following section; the companding scheme for bit-width reduction in each butterfly is described below.

---

**Figure 3.1.1** Block diagram of MDCT core
3.2.1 1-D DCT design to minimize arithmetic operations
Our 1-D DCT for minimum arithmetic operations is based on Chen’s factorization that decomposes the 8x1 DCT to twelve 2-input butterflies, as shown in Fig. 1. Each butterfly

![Butterfly representing a 8x8 matrix multiplication](image)

Represents a 2x2 matrix multiplication; six butterflies, without all constants being unity, are generally computed with four multiplications, one addition and one subtraction. Although Wang’s factorization [9] reduces these operations to three multiplications, two additions and one subtraction, Chen’s algorithm is expected to eliminate more redundant additions among the four multiplications due to the symmetry of the constant pair. In most of the 2-D DCT implementations for image/video coders, 12-bit precision has been used in order to conform to IEEE 1180-1990 accuracy specifications.

**Table3.2.1** Unsigned 11-bit expressions of the cosine coefficients for the 8x1 DCT

<table>
<thead>
<tr>
<th>Co.DCT-II and DCT/DCT-III</th>
<th>Decimal</th>
<th>Unsigned 11-bit binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cos(π/4)</td>
<td>0.70710678</td>
<td>0.10110101000₂</td>
</tr>
<tr>
<td>DCT-IV</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cos(π/8)</td>
<td>0.92387953</td>
<td>0.11101100100₂</td>
</tr>
<tr>
<td>Cos(3π/8)</td>
<td>0.83268343</td>
<td>0.01100010000₂</td>
</tr>
<tr>
<td>Butterfly I</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cos(π/16)</td>
<td>0.98078528</td>
<td>0.11111011000₂</td>
</tr>
<tr>
<td>Cos(7π/16)</td>
<td>0.19509032</td>
<td>0.00110010000₂</td>
</tr>
<tr>
<td>Butterfly II</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cos(3π/16)</td>
<td>0.83146961</td>
<td>0.11010100111₂</td>
</tr>
<tr>
<td>Cos(5π/16)</td>
<td>0.555557023</td>
<td>0.100011100100₂</td>
</tr>
</tbody>
</table>

4. IMPLEMENTATION
The hardware chosen for implementation of the DCT algorithms is the Field Programmable Gate Array (FPGA) which has widespread use in the development of Application Specific Integrated Circuits (ASICs). The design is implemented on Virtex4 family FPGA using Xilinx ISE. We designed all the design blocks using Verilog module.

4.1 COMPARISION WITH CONVENTIONAL AND PROPOSED ARCHITECTURE:
We implemented the proposed design and the conventional multiplier less DCT architectures NEDA and CORDIC on the Xilinx XC2VP50 FPGA; we compared and measured the power consumption with the XPower tool. The required
area of the proposed design without the companding scheme is shown in Table III; the area was measured in number of slices.

Table 4.1(a) the performance of various Multiplier less 8×1 DCT Architectures

<table>
<thead>
<tr>
<th></th>
<th>NEDA</th>
<th>CORDIC</th>
<th>Proposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st DCT</td>
<td>1031</td>
<td>780</td>
<td>531</td>
</tr>
<tr>
<td>2nd DCT</td>
<td>1447</td>
<td>951</td>
<td>697</td>
</tr>
<tr>
<td>TOTAL</td>
<td>2478</td>
<td>1731</td>
<td>1228</td>
</tr>
<tr>
<td></td>
<td>(100%)</td>
<td>(69.8%)</td>
<td>(49.5%)</td>
</tr>
</tbody>
</table>

Two popular test images, Lena and Mandrill, were used to estimate the power consumption. All designs were operated at 50MHz and 1.5 Volt. Each of these images has 512×512 pixels, with each pixel being represented by 8 bits for a total of 256 gray levels. The energy consumption is summarized in Table IV. The results show that the proposed architecture reduces the power dissipation by up to 90.0% compared to the conventional NEDA.

Table 4.1(b) Synthesis results for the proposed architecture and the conventional Neda and Cordic.

<table>
<thead>
<tr>
<th>Images</th>
<th>NEDA</th>
<th>CORDIC</th>
<th>Proposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>LENA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1st DCT</td>
<td>261.5</td>
<td>29.4</td>
<td>23.2</td>
</tr>
<tr>
<td>2nd DCT</td>
<td>389.5</td>
<td>52.3</td>
<td>40.5</td>
</tr>
<tr>
<td>TOTAL</td>
<td>651.0</td>
<td>81.7</td>
<td>63.7</td>
</tr>
<tr>
<td></td>
<td>(100%)</td>
<td>(12.6%)</td>
<td>(9.78%)</td>
</tr>
<tr>
<td>MANDRIL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1st DCT</td>
<td>270.2</td>
<td>33.7</td>
<td>26.6</td>
</tr>
<tr>
<td>2nd DCT</td>
<td>400.3</td>
<td>52.2</td>
<td>40.6</td>
</tr>
<tr>
<td>TOTAL</td>
<td>670.5</td>
<td>85.9</td>
<td>67.2</td>
</tr>
<tr>
<td></td>
<td>(100%)</td>
<td>(12.8%)</td>
<td>(10.02%)</td>
</tr>
</tbody>
</table>

The reduced bits are determined by considering the bit distribution in table 4.1(b). Our scheme can reduce the power consumption by up to 15% while PSNR degrades by less than 12dB. More power savings are expected for ASIC implementations of our scheme. This is because our results come from an FPGA implementation using lookup tables without signal transition during an arithmetic operation.

5. SIMULATION

5.1 Simulation Results

Self-verifying test bench is written which takes mat lab-converted image as input. Core transforms it to DCT coefficients and behavioral IDCT test bench code reconstructs from its original image. Mat lab scripts are written for converting 8 bit bitmap to txt format readable by test bench and vice versa are also written. You can see here original image and 2D-DCT core-transformed image.
The program code written in Verilog for multiplier-less DCT is verified in Xilinx ISE simulator for observing the output waveform. The output will be in the form of hexadecimal values. This is obtained when the pixel values of the image taken from the MATLAB is being saved as inputs in the memory locations. This is then undergone transformation called Discrete Cosine Transformation. This output is verified with the DCT output in the MATLAB for conformation (not necessary).

![Figure 5.1(a) Original Image of Lena](image1)

![Figure 5.1(b) 2D-DCT image of Lena](image2)

![Figure 5.1(c) The Simulation Waveform](image3)

![Figure 5.1(d) The schematic diagram is as shown in the above fig which indicates the input, clock signals reset and output signals.](image4)
6. CONCLUSION

Performance: 512 x 512 x 8 bit image is 2D DCT transformed in about 26.3 ms with 10 MHz input clock. This gives 10 mega samples per second throughput with this frequency. When the pipeline is full new 2D DCT data is output on every clock cycle. Core latency is 85 clock cycles (time for latching first input to giving first DCT output). Area: 1621 slices and 2 RAMB16s Block rams on Xilinx Virtex4 xc4vsx35 device (99 MHz max freq) with synchronous on chip ROM.

References