A High-Accuracy and High-Speed 2-D 8x8 Discrete Cosine Transform Design

Trang T.T. Do, Binh P. Nguyen
Department of Electrical and Computer Engineering
National University of Singapore, Singapore 117576
{dothutrang, phubinh}@nus.edu.sg

Abstract—In this paper, a high-accuracy and high-speed 2-D 8x8 Discrete Cosine Transform (DCT) design is proposed. By using the parallel architecture, inherited from the design reported by S. Ramachandran et. al., re-designing and enhancing function of all component modules, the proposed design can transform each 8x8 block in 64 cycles and achieve a very high accuracy compared to other designs and standard references. The proposed design implemented on Xilinx Virtex 4 can run at the speed of 308 MHz, processing images of size 1920 x 1080 pixels at the rate of 145 frames per second.

Keywords- DCT, Discrete Cosine Transform, Pipeline, FPGA, Parallel Matrix Multiplication, Parallel Architecture.

I. INTRODUCTION

The Two Dimensional Discrete Cosine Transform (2-D DCT) is used as a transform coding scheme in most video standards such as HDTV video coding, JPEG and MPEG. Since DCT requires a highly intensive computation, different efficient DCT algorithms have been reported. There have also been numerous efforts to efficiently map and implement these algorithms into VLSI chips. In the literature, some hardware designs have a very high speed but low accuracy, while some other designs have a good accuracy but running at a low speed. Therefore, an efficient design running at a high speed and having a high accuracy is needed.

Among the high-speed group, the DCT processor in [1]-[3] is one of the fastest designs. It is implemented based on a parallel matrix multiplication algorithm and a parallel architecture. However, it is not as accurate as standard references such as Matlab (after rounding), and Xdiv MPEG 4.

Inherited from this parallel architecture, a high-accuracy and high-speed DCT design, which can transform each 8x8 image block in 64 cycles, is proposed. By efficiently re-designing all modules, the proposed design has a significant improvement in terms of accuracy while still maintaining the high speed compared to the original architecture and other designs.

The remaining of the paper is organized as follows. The parallel algorithm in [1]-[3] is represented in Section II. The design of the DCT module is proposed in Section III, followed by the Simulation Results and Discussion in Section IV. The paper ends with Conclusion in Section V.

II. PARALLEL MATRIX MULTIPLICATION ALGORITHM FOR DCT

2D-DCT for an 8x8 pixels block of a digital image is defined as:

\[
DCT(u,v) = \frac{1}{4} c(u)c(v) \sum_{i=0}^{7} \sum_{j=0}^{7} f(i,j) \left[ \cos \left( \frac{(2x+1)\pi}{16} \right) \cos \left( \frac{(2y+1)\pi}{16} \right) \right] \quad (1)
\]

where

\[
f(x,y) \text{ is the pixel intensity}
\]

\[
c(u) = c(v) = \frac{1}{\sqrt{2}} \quad \text{for } u = v = 0 \text{ and}
\]

\[
= 1 \quad \text{for } u, v \in [1,7]
\]

The 2-D DCT can be conveniently expressed in a matrix form:

\[
DCT = XCX^T
\]

where \(X\) is the input image matrix, \(C\) and \(C^T\) are the cosine coefficient matrix and its transpose with constants \(\frac{1}{2} c(u)\) and \(\frac{1}{2} c(v)\) absorbed in \(C\) and \(C^T\) respectively.

An expanded form of (3) is as followed:

\[
DCT = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

where

\[
P_{ik} = \sum_{j=0}^{7} P_{jk} c_{ik}
\]

Equation (3) can be implemented by parallel architecture wherein eight partial products (row vectors of
matrix \( CX \) generated in the first stage are fed to the second stage in a pipeline. Subsequently, eight row vectors of \( CX \) are multiplied by the \( C^T \) matrix to generate eight DCT coefficients which correspond to a row of \( CXC^T \). While computing the \((i+1)\)th partial products of \( CX \), the \( i \)th row DCT coefficients can also be computed simultaneously since the \( i \)th partial products of \( CX \) are already available.

III. THE PROPOSED DCT PROCESSOR

A. DCT Core Top Level

The proposed DCT processor architecture is inherited from the parallel architecture of [1]-[3], which includes a controller, a datapath, a ROM and a RAM (Fig. 1). All the modules are re-designed with enhanced function to increase the accuracy.

The input/output pin system of DCT processor is shown in Fig. 1a. Any image block (8 x 8 pixels) can be input as one row of a block, i.e., 8 pixels at a time; via a 64-bit bus. The DCT processor takes 64 clock cycles to process one block of image. Assuming that the input bus clock, “pci clk”, and the DCT clock, “clk”, are the same, the host has enough free time (56 clock cycles) to attend to other processing cores. The data input bus is denoted as “\( di[63:0] \)”. Further, be[7:0] is used to identify which bytes in “\( di \)” are enabled. Also, it is necessary to have 3 bits of write address, “\( wa[2:0] \)”, for writing eight rows of a block of data. All these activities must be synchronous to the clock signal, “pci clk”. When the data input, “\( di \)”, is valid, the DCT engine is informed by a signal “\( \text{din_valid} \)”. Besides, an asynchronous, active low signal, “\( \text{reset_n} \)” is used to reset the system. The description of DCT core signal can be seen in Table I.

B. Sequence of Operations

The operation sequences of the host processor and the DCT processor are as follows:

For the host:

1. Assert “\( \text{reset_n} \)” signal to initialize the DCT processor.

2. After ascertaining that “\( \text{ready} \)” signal is set by DCT processor, write an image block of 8 x 8 pixels into one block of the 64-byte Dual RAM in the DCT core via the data bus “\( \text{di}[63:0] \)”, 8 bytes each time. During this data input duration, “\( \text{din_valid} \)” is asserted as well. Besides, the “\( \text{be} \)” bits may be simultaneously asserted.

3. Issue “\( \text{start} \)” signal to begin DCT processing. This signal must be continuously asserted for continuous processing.

4. Wait for the set “\( \text{ready} \)” signal. Then write the next image block into another block of 64-byte Dual RAM in the DCT core.

Note: Normally, the host processor asserts “\( \text{reset_n} \)” signal once at the beginning of the DCT processing. However, it may apply “\( \text{reset_n} \)” anytime to terminate the current processing.

For the DCT processor:

1. If “\( \text{reset_n} \)” is active, the core will terminate the current processing and initialize all internal registers. One bank of the 64-byte Dual RAM is in write-only mode for the host to write the image block into, while the other is in read-only mode for the DCT processing. Set the “\( \text{ready} \)” signal to inform the host about its ready status.

2. Wait for the asserted “\( \text{start} \)” signal to begin the processing. Assert “\( \text{ready} \)” signal for the host to write image data into the other Dual RAM bank concurrently. After being computed, the DCT coefficients are issued at dct[11:0] pins, valid at the positive edge of “\( \text{clk} \)” with “\( \text{dct_valid} \)” signal asserted after a latency of 37 cycles. “\( \text{dct_valid} \)” signal is continuously asserted as long as the processing continues without a break. “\( \text{addr}[5:0] \)” is valid upon the DCT is valid. This step is continuously processed as long as “\( \text{start} \)” is kept asserted and “\( \text{reset_n} \)” or “\( \text{hold} \)” is not active.

<table>
<thead>
<tr>
<th>Signal</th>
<th>In/Out</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clk</td>
<td>Input</td>
<td>DCT system clock.</td>
</tr>
<tr>
<td>pci_clk</td>
<td>Input</td>
<td>Bus clock used to transfer data to DCT system.</td>
</tr>
<tr>
<td>be[7:0]</td>
<td>Input</td>
<td>Byte enable active low signal, be[0] selects di[7:0]</td>
</tr>
<tr>
<td>wa[2:0]</td>
<td>Input</td>
<td>This furnishes the row address of an image block.</td>
</tr>
<tr>
<td>wa[0]</td>
<td>Input</td>
<td>wa[0] is the first row.</td>
</tr>
<tr>
<td>din[63:0]</td>
<td>Input</td>
<td>This furnishes the row address of an image block.</td>
</tr>
<tr>
<td>din_valid</td>
<td>Input</td>
<td>din_valid is the first pixel and di[63:56] is the last in the row. A pixel is of size 8 bits, unsigned.</td>
</tr>
<tr>
<td>d</td>
<td>Input</td>
<td>This input signals when the input data, di[63:0], is valid.</td>
</tr>
<tr>
<td>start</td>
<td>Input</td>
<td>Active high starts and maintains the DCT processing.</td>
</tr>
<tr>
<td>hold</td>
<td>Input</td>
<td>DCT processing can be kept on hold. De-asserting resumes the processing without any latency.</td>
</tr>
<tr>
<td>reset_n</td>
<td>Input</td>
<td>Asynchronous, active low.</td>
</tr>
<tr>
<td>ready</td>
<td>Output</td>
<td>Indicates the ready status to accept image input block of the core.</td>
</tr>
<tr>
<td>dct_valid</td>
<td>Output</td>
<td>Indicates the validity of DCT coefficient.</td>
</tr>
<tr>
<td>addr[5:0]</td>
<td>Output</td>
<td>DCT coefficient address</td>
</tr>
<tr>
<td>det[11:0]</td>
<td>Output</td>
<td>DCT output in 2-complement</td>
</tr>
</tbody>
</table>

Note: Normally, the host processor asserts “\( \text{reset_n} \)” signal once at the beginning of the DCT processing. However, it may apply “\( \text{reset_n} \)” anytime to terminate the current processing.
the rising edge of "clk".

matrix multiplication CX. "ra[2:0]" is the read address to
X matrix is accessed by column when computing the
used to configure one of the two RAMs in write mode
positive edge of "pci_clk" signal. The signal, "rnw",
eight 64-bits-width locations. Image data is written at the
block, we need 3 bits address, wa[2:0], corresponding to
of the 8 bits in the 64 bit data bus. For writing one pixel
writing into, the first one will be read concurrently for
RAMs is filled and once it is full, the image data is written
algorithm described in Section II. Initially, one of the two
stores the image data written from a host processor.
matrices. These two matrices are used instead of C and C
3) Datapath
In this module (Fig. 2), eight multipliers, “mult16sx8u”, accomplish the computation of CX, where
X is the unsigned input and C is the signed cosine term. These results are in 2-complement. The next module, “adder24s” is used for adding 24-bit partial products of CX (result1–result8). After being cut off the last 11 bits, the result, “sum1” is stored in “dctreg2x8xn” registers, which contains eight numbers of 16-bit registers, qr0–qr7. In order to compute (CX)C, each row of partial products of CX is stored in these registers and will be preserved for
the next eight “clk” cycles. In the second stage, the partial
products, qr0–qr7 are taken to multiply with the corresponding column elements of Cᵣ for producing independent multiplied 32-bit results “res1”–“res8”. The eight multipliers, “mult16sx16s”, are used to accomplish
this. Note that the bit precision has increased. “dout2” of
ROM module is the Cᵣ output, fed as one of the inputs to
these multipliers. These results, “res1” through “res8”, are
added together by “adder29sr” module. Note that we
retain only the most significant 29 bits for each input of
this adder. The sum, naturally, is 18 bits more. The least
significant 18 bits and the most significant 2 bits of the
result are truncated, retaining only 12 bits. It is a DCT
coefficient.

4) Controller
This module generates all the control and handshake
signals required for other modules to compute DCT
coefficients. There are four counters, “cnt1_reg”, “cnt2_reg”, “cnt3_reg” and “addr” in the controller
design. The first three counters are the pipeline registers
and counter “addr” serves as the address (index) of the
DCT coefficients. The enable signals of the four counters,
“encnt1” to “encnt4” are realized by four “always”
sequential blocks in Verilog. Note that all the “always”
sequential blocks have their respective signals initialized
when system reset “reset_n” is applied and the signals’
value are frozen so long as “hold” signal is asserted. The
first counter is enabled right at the beginning when the user
asserts the “start” signal (after reset signal is withdrawn) to commence the DCT computation. And it is
disabled when DCT engine completes processing the last
coefficient of a block. The enable signals, “encnt2” to
“encnt4”, are activated when the first counter “cnt1_reg” is
respectively 14, 20 and 36 in order to allow the counters
“cnt2_reg”, “cnt3_reg” and “addr” to start running at the
appropriate time. “cnt1_reg” to “cnt3_reg” serve as the
read addresses for the modules, “dualram”, “romc” and
“dctreg2x8xn”, and “addr” serves as the address of the
DCT coefficients.

IV. SIMULATION RESULTS AND DISCUSSION
The proposed design can be implemented on FPGAs or ASICs. In simulation phase, a single Xilinx FPGA
device, XC4VLX25-12SF363 was used. The simulation
shows that the latency between input and output of this
DCT core is 37 cycles. Once the pipeline (37-depth) is
full, the DCT coefficients can be issued one every clock
cycle without a break. It takes 101 cycles totally to
process the first image block. However, since the second
block onward, as long as the “start” signal is kept asserted
and “reset_n” or “hold” is not active, it will take only 64
cycles for processing each image block. This also can be
derived from the architecture and the operation of the
DCT core.

In this FPGA implementation, the maximum
frequency achieved is 308 MHz (Table 3), meaning that
each cycle equals to 3.3 ns. DCT computation for an 8 x 8
pixel block requires 64 clock cycles, that is, 212 ns.
Therefore, for a HDTV 1920 x 1080 pixel image frame
(32400 blocks), the processing time will be approximately
6.9 ms (32400 x 212 ns). Hence, it is capable of
processing images of size 1920 x 1080 pixels at the rate of
145 frames per second.

In order to compare the simulation results with the
original design, another DCT core architecture extracted
from the DCTQ processor design in [3] was implemented
as well. The synthesis results of the two designs can be
summarized in Table II and Table III.

While the new design appears to be more resource
demanding compared to the original design, it does not
pose any implementation problem. In fact, the remaining
resource is more than sufficient to implement cores like
quantization and Huffman coding. As for the throughput,
both the two designs provide extremely high speed
performance and capable of real-time operations. The key
advantage in the new design is its accuracy. Several tests
are carried out, showing that the original architecture’s
results deviate from Matlab computed values. The same
tests on the new architecture produce DCT coefficients
that are the same as that calculated by Matlab. Table 4
compares the first 16 DCT coefficients of an image block
produced by the Matlab, the new design, the original
design, and a 2-D DCT implementation of [4], which was
used in the source code of Xvid MPEG-4 Video Codec.
All bold values are different from the Matlab values after
being rounded.

In this DCT processor architecture, while the input
data is read by block, the output DCT coefficients are
produced serially. This may slow down the processing
speed of the next component. However, this problem can
easily be solved by adding one Dual RAM module to the
design. This RAM will be used as a buffer for storing the
DCT coefficients.

V. CONCLUSIONS

Based on the reported parallel architecture [1]-[3], the
proposed design with the enhancement in all component
modules can transform each 8x8 block in 64 cycles and
achieve a very high accuracy while maintaining the speed.
The proposed design can run at the maximum speed of
308 MHz on Xilinx Virtex 4 platform, processing images
of size 1920 x 1080 pixels at the rate of 145 frames per
second, which meets the real-time requirements for high
definition. Moreover, this architecture has a great
potential in practical since the matrix multiplication form
CXC\(^2\) is frequently used in numerous problems.

REFERENCES

implementation of 2D discrete cosine transform using EPLDs”,

Architecture of Real Time 2D-Discrete Cosine Transform and
Quantization for Image Compression”, IEEE International
Symposium on Circuits and Systems (ISCAS ’99), Orlando,

implementation of an MPEG based video scalar with reduced on-
51, no. 6-7, pp. 435-450, 2005.

DCT Algorithms with 11 Multiplications”, Proc. Int'l. Conf. on
988-991.