XAPP610 (v1.2) April 24, 2002

Video Compression Using DCT

Author: Latha Pillai

Summary

This application note describes a two-dimensional Discrete Cosine Transform (2D DCT) function implemented on a Xilinx FPGA. The reference design file provides behavioral code for implementation on any Xilinx device. Some advantages of the module include the ability to parametrize the DCT function and to guarantee performance. The code can be further optimized by instantiating embedded adders and multipliers when targeting the Virtex™-II series of FPGAs. After an initial latency of 92 clock cycles, one 2D-DCT value is output at every clock.

Compression

Compression is the process of reducing the size of the data sent, thereby, reducing the bandwidth required for the digital representation of a signal. Many inexpensive video and audio applications are made possible by the compression of signals. Compression technology can result in reduced transmission time due to less data being transmitted. It also decreases the storage requirements because there is less data. However, signal quality, implementation complexity, and the introduction of communication delay are potential negative factors that should be considered when choosing compression technology.

Video and audio signals can be compressed because of the spatial, spectral, and temporal correlation inherent in these signals. Spatial correlation is the correlation between neighboring samples in an image frame. Temporal refers to correlation between samples in different frames but in the same pixel position. Spectral correlation is the correlation between samples of the same source from multiple sensors.

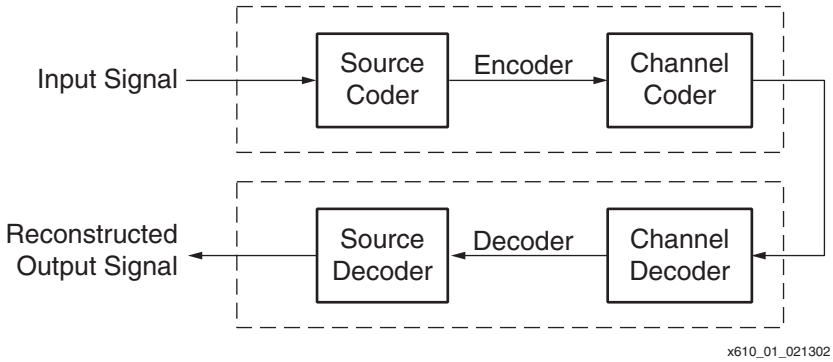

Figure 1: **Block Diagram of a Compression/Decompression System[1]**

There are two categories of compression: lossy and lossless. In medical system applications, image losses can translate into costly medical mistakes; therefore, lossless compression methods are used. Fortunately, the majority of video and image processing applications do not require the reconstructed data to be identical to the original data. In such applications, lossy compression schemes can be used to achieve higher compression ratios.

DCT Compression

Discrete Cosine Transform (DCT) is a lossy compression scheme where an N x N image block is transformed from the spatial domain to the DCT domain. DCT decomposes the signal into spatial frequency components called DCT coefficients. The lower frequency DCT coefficients appear toward the upper left-hand corner of the DCT matrix, and the higher frequency coefficients are in the lower right-hand corner of the DCT matrix. The Human Visual System (HVS) is less sensitive to errors in high frequency coefficients than it is to lower frequency coefficients. Because of this, the higher frequency components can be more finely quantized, as done by the quantization matrix.

Each value in the quantization matrix is pre-scaled by multiplying by a single value, known as the quantizer scale code. This value can range in value from one to 112 and is modifiable on a macroblock basis. Dividing each DCT coefficient by an integer scale factor and rounding the results accomplishes quantization. This sets the higher frequency coefficients (in the lower right corner), that are less significant to the compressed picture, to zero by quantizing in larger steps. The low frequency coefficients (in the upper left corner), that are more significant to the compressed picture, are quantized in smaller steps. The goal of quantization is to force as many of the DCT coefficients to zero, or near zero, as possible within the boundaries of the prescribed bit-rate and video quality parameters. Thus, since quantization throws away some information, it is a lossy compression scheme.

Quantization can be expressed as:

$$Quantizedvalue_{(i,j)} = \frac{DCT_{(i,j)}}{Quantizationmatrix_{(i,j)}}$$

The following is a sample of the Quantization matrix used for the JPEG algorithm:

$$\text{JPEG Quantization Matrix} = \begin{bmatrix} 3 & 5 & 7 & 9 & 11 & 13 & 15 & 17 \\ 5 & 7 & 9 & 11 & 13 & 15 & 17 & 19 \\ 7 & 9 & 11 & 13 & 15 & 17 & 19 & 21 \\ 9 & 11 & 13 & 15 & 17 & 19 & 21 & 23 \\ 11 & 13 & 15 & 17 & 19 & 21 & 23 & 25 \\ 13 & 15 & 17 & 19 & 21 & 23 & 25 & 27 \\ 15 & 17 & 19 & 21 & 23 & 25 & 27 & 29 \\ 17 & 19 & 21 & 23 & 25 & 27 & 29 & 31 \end{bmatrix}$$

For most image compression standards, N = 8. An 8 x 8 block size does not have significant memory requirements, and furthermore, a block size greater than 8 x 8 does not offer significantly better compression. Each picture is divided into 16 x 16 blocks called a macroblock. Each of the macroblocks is further divided into 16 samples x 16 lines. Each block on which DCT is performed is 8-samples by 8-lines called blocks. Thus, each macroblock consists of four blocks.

DCT is image independent and can be performed with fast algorithms. Fast algorithms are parallel and can be efficiently implemented on parallel architecture. Examples of standards using DCT:

- Dolby AC2 & AC3: 1-D DCT (and 1-D Discrete Sine Transform)
- JPEG (still images): 2-D DCT spatial compression
- MPEG1 & MPEG2: 2-D DCT plus motion compensation
- H.261 and H.263: moving image compression for video conferencing and video telephony

Much of the processing required to encode or decode video using these standards is taken up by calculating the DCT and/or IDCT. An efficient hardware block dedicated to these functions will improve the performance of the digital video system considerably.

Technical Aspects of the Compression Algorithm

The following material provides a brief description of DCT. Further details can be found in "Image and Video Compression Standards" by Vasudev Bhaskaran and Konstantinos Konstantinides[1]. The 12-bit signed input pixels provide a 16-bit signed output coefficient for the DCT. A 12-bit signed input has 11 data bits. For an 8 x 8 DCT, the 11 data bits can be multiplied eight times (represented by three bits). Multiplying 11 bits by 3 bits can result in 11 + 3 or 14 bits. Added to these 14 bits are the sign bit and the fraction bit to give a total of 16 bits. The algorithm used for the calculation of the 2D DCT is based on the following equation:

$$XC_{pq} = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} XN_{mn} \cdot \frac{c(p)c(q)}{4} \cdot \cos\frac{\pi(2m+1)p}{2M} \cdot \cos\frac{\pi(2n+1)q}{2N} \quad \text{(EQ 1)}$$

First, the 1D DCT of the rows are calculated and then the 1D DCT of the columns are calculated. The 1D DCT coefficients for the rows and columns can be calculated by separating equation 1 into the row part and the column part.

$$C = K \bullet \cos\frac{(2 \cdot \text{col number} + 1) \bullet \text{row number} \bullet \pi}{2 \bullet M} \quad \text{(EQ 2), where}$$

$$K = \frac{\sqrt{1}}{N} \quad \text{for} \quad \text{row} = 0$$

$$K = \frac{\sqrt{2}}{N} \quad \text{for} \quad \text{row} \neq 0$$

$$C^t = K \bullet \cos\frac{(2 \cdot \text{row number} + 1) \bullet \text{col number} \bullet \pi}{2 \bullet N} \quad \text{(EQ 3), where}$$

$$K = \frac{\sqrt{1}}{M} \quad \text{for} \quad \text{col} = 0$$

$$K = \frac{\sqrt{2}}{M} \quad \text{for} \quad \text{col} \neq 0$$

and M = total # of columns, N = total # of rows.

The constant values for C and C^t calculated from equations 2 and 3 are as follows:

$$C = \begin{bmatrix} 23170 & 23170 & 23170 & 23170 & 23170 & 23170 & 23170 & 23170 \\ 32138 & 27246 & 18205 & 6393 & -6393 & -18205 & -27246 & -32138 \\ 30274 & 12540 & -12540 & -30274 & -30274 & -12540 & 12540 & 30274 \\ 27246 & -6393 & -32138 & -18205 & 18205 & 32138 & 6393 & -27246 \\ 23170 & -23170 & -23170 & 23170 & 23170 & -23170 & -23170 & 23170 \\ 18205 & -32138 & 6393 & 27246 & -27246 & -6393 & 32138 & -18205 \\ 12540 & -30274 & 30274 & -12540 & -12540 & 30274 & -30274 & 12540 \\ 6393 & -18205 & 27246 & -32138 & 32138 & -27246 & 18205 & -6393 \end{bmatrix}$$

$$C^t = \begin{bmatrix} 23170 & 32138 & 30274 & 27246 & 23170 & 18205 & 12540 & 6393 \\ 23170 & 27246 & 12540 & -6393 & -23170 & -32138 & -30274 & -18205 \\ 23170 & 18205 & -12540 & -32138 & -23170 & 6393 & 30274 & 27246 \\ 23170 & 6393 & -30274 & -18205 & 23170 & 27246 & -12540 & -32138 \\ 23170 & -6393 & -30274 & 18205 & 23170 & -27246 & -12540 & 32138 \\ 23170 & -18205 & -12540 & 32138 & -23170 & -6393 & 30274 & -27246 \\ 23170 & -27246 & 12540 & 6393 & -23170 & 32138 & -30274 & 18205 \\ 23170 & -32138 & 30274 & -27246 & 23170 & -18205 & 12540 & -6393 \end{bmatrix}$$

For the case of 8 x 8 block region, a one-dimensional 8-point DCT/IDCT followed by an internal double buffer memory, followed by another one-dimensional 8-point DCT provides the 2D DCT architecture.

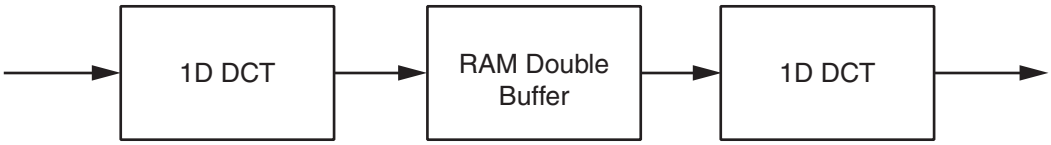

Figure 2: **2D-DCT Using Vector Processing**

Vector processing using parallel multipliers is a method used for implementation of DCT. The advantages in the vector processing method are regular structure, simple control and interconnect, and good balance between performance and complexity of implementation.

Behavioral Model

Using vector processing, the output Y of an 8 x 8 DCT for input X is given by $Y = C \bullet X \bullet C^t$, where C is the cosine coefficients and C^t are the transpose coefficients. This equation can also be written as $Y=C \bullet Z$, where $Z = X \bullet C^t$. Using the cosine C and inverse cosine C^t numbers in equation 2 and 3, the intermediate value $Z = X \bullet C^t$ can be calculated as follows:

$$X = \begin{bmatrix} x00 & x01 & x02 & x03 & x04 & x05 & x06 & x07 \\ x10 & x11 & x12 & x13 & x14 & x15 & x16 & x17 \\ x20 & x21 & x22 & x23 & x24 & x25 & x26 & x27 \\ x30 & x31 & x32 & x33 & x34 & x35 & x36 & x37 \\ x40 & x41 & x42 & x43 & x44 & x45 & x46 & x47 \\ x50 & x51 & x52 & x53 & x54 & x55 & x56 & x57 \\ x60 & x61 & x62 & x63 & x64 & x65 & x66 & x67 \\ x70 & x71 & x72 & x73 & x74 & x75 & x76 & x77 \end{bmatrix} \quad C^t = \begin{bmatrix} 23170 & 32138 & 30274 & 27246 & 23170 & 18205 & 12540 & 6393 \\ 23170 & 27246 & 12540 & -6393 & -23170 & -32138 & -30274 & -18205 \\ 23170 & 18205 & -12540 & -32138 & -23170 & 6393 & 30274 & 27246 \\ 23170 & 6393 & -30274 & -18205 & 23170 & 27246 & -12540 & -32138 \\ 23170 & -6393 & -30274 & 18205 & 23170 & -27246 & -12540 & 32138 \\ 23170 & -18205 & -12540 & 32138 & -23170 & -6393 & 30274 & -27246 \\ 23170 & -27246 & 12540 & 6393 & -23170 & 32138 & -30274 & 18205 \\ 23170 & -32138 & 30274 & -27246 & 23170 & -18205 & 12540 & -6393 \end{bmatrix}$$

$$Z_{(0,0)} = 23170\ (x_{00} + x_{01} + x_{02} + x_{03} + x_{04} + x_{05} + x_{06} + x_{07})$$

$$Z_{(0,1)} = 32138x_{00} + 27246x_{01} + 18205x_{02} + 6393x_{03} - 6393x_{04} - 18205x_{05} - 27246x_{06} - 32138x_{07})$$

$$= 32138(x_{00} - x_{07}) + 27246(x_{01} - x_{06}) + 18205(x_{02} - x_{05}) + 6393(x_{03} - x_{04})$$

$$Z_{(0,2)} = 30274(x_{00} + x_{07}) + 12540(x_{01} + x_{06}) - 12540(x_{02} + x_{05}) - 30274(x_{03} + x_{04})$$

$$Z_{(0,3)} = 27246(x_{00} - x_{07}) - 6393(x_{01} - x_{06}) - 32138(x_{02} - x_{05}) - 18205(x_{03} - x_{04})$$

$$Z_{(0,4)} = 23170(x_{00} + x_{07}) - 23170(x_{01} + x_{06}) - 23170(x_{02} + x_{05}) + 23170(x_{03} + x_{04})$$

$$Z_{(0,5)} = 18205(x_{00} - x_{07}) - 32138(x_{01} - x_{06}) + 6393(x_{02} - x_{05}) + 27246(x_{03} - x_{04})$$

$$Z_{(0,6)} = 12540(x_{00} + x_{07}) - 30274(x_{01} + x_{06}) + 30274(x_{02} + x_{05}) - 12540(x_{03} + x_{04})$$

$$Z_{(0,7)} = 6393(x_{00} - x_{07}) - 18205(x_{01} - x_{06}) + 27246(x_{02} - x_{05}) - 32138(x_{03} - x_{04})$$

Or:

$$Z_{(k,0)} = 23170\ (x_{k0} + x_{k1} + x_{k2} + x_{k3} + x_{k4} + x_{k5} + x_{k6} + x_{k7})$$

$$Z_{(k,1)} = 32138(x_{k0} - x_{k7}) + 27246(x_{k1} - x_{k6}) + 18205(x_{k2} - x_{k5}) + 6393(x_{k3} - x_{k4})$$

$$Z_{(k,2)} = 30274(x_{k0} + x_{k7}) + 12540(x_{k1} + x_{k6}) - 12540(x_{k2} + x_{k5}) - 30274(x_{k3} + x_{k4})$$

$$Z_{(k,3)} = 27246(x_{k0} - x_{k7}) - 6393(x_{k1} - x_{k6}) - 32138(x_{k2} - x_{k5}) - 18205(x_{k3} - x_{k4})$$

$$Z_{(k,4)} = 23170(x_{k0} + x_{k7}) - 23170(x_{k1} + x_{k6}) - 23170(x_{k2} + x_{k5}) + 23170(x_{k3} + x_{k4})$$

$$Z_{(k,5)} = 18205(x_{k0} - x_{k7}) - 32138(x_{k1} - x_{k6}) + 6393(x_{k2} - x_{k5}) + 27246(x_{k3} - x_{k4})$$

$$Z_{(k,6)} = 12540(x_{k0} + x_{k7}) - 30274(x_{k1} + x_{k6}) + 30274(x_{k2} + x_{k5}) - 12540(x_{k3} + x_{k4})$$

$$Z_{(k,7)} = 6393(x_{k0} - x_{k7}) - 18205(x_{k1} - x_{k6}) + 27246(x_{k2} - x_{k5}) - 32138(x_{k3} - x_{k4})$$

where k = 0, 2, …, 7.

Each element in the first row of the input matrix X are multiplied by each element in the first column of matrix C^t and added together to get the first value Z_{00} of the intermediate matrix Z. To get Z_{01}, each element of row zero in X is multiplied by each element in the first column of C^t and added and so on. As shown, input X_{00} gets multiplied by all the coefficients in the first row of C^t, and input X_{01} gets multiplied by all the coefficients in the second row of C^t, and so on.

The calculation is implemented by using eight multipliers and storing the coefficients in ROMs. At the first clock, the eight inputs x_{00} to x_{07} are multiplied by the eight values in column one, resulting in eight products (P_{00_0} to P_{00_7}). At the eighth clock, the eight inputs are multiplied by the eight values in column eight resulting in eight products (P_{07_0} to P_{07_7}). From the equations for Z, the intermediate values for the first row of Z is computed:

$$Z_{00} = P_{00\_0} + P_{00\_1} + P_{00\_2} + P_{00\_3} + P_{00\_4} + P_{00\_5} + P_{00\_6} + P_{00\_7}$$

$$Z_{01} = P_{01\_0} + P_{01\_1} + P_{01\_2} + P_{01\_3} + P_{01\_4} + P_{01\_5} + P_{01\_6} + P_{01\_7}$$

$$Z_{ij} = P_{ij\_0} + P_{ij\_1} + P_{ij\_2} + P_{ij\_3} + P_{ij\_4} + P_{ij\_5} + P_{ij\_6} + P_{ij\_7}$$

Where i = 0 to 7, the matrix X row number and j = 0 to 7, the matrix C^t column number.

The intermediate values for the second row of Z, Z_{1K}, are computed by using P_{1K} values. The 8 x 8 Z matrix can be calculated using these equations.

The block diagram for the implementation of the 1D-DCT is shown in Figure 3.

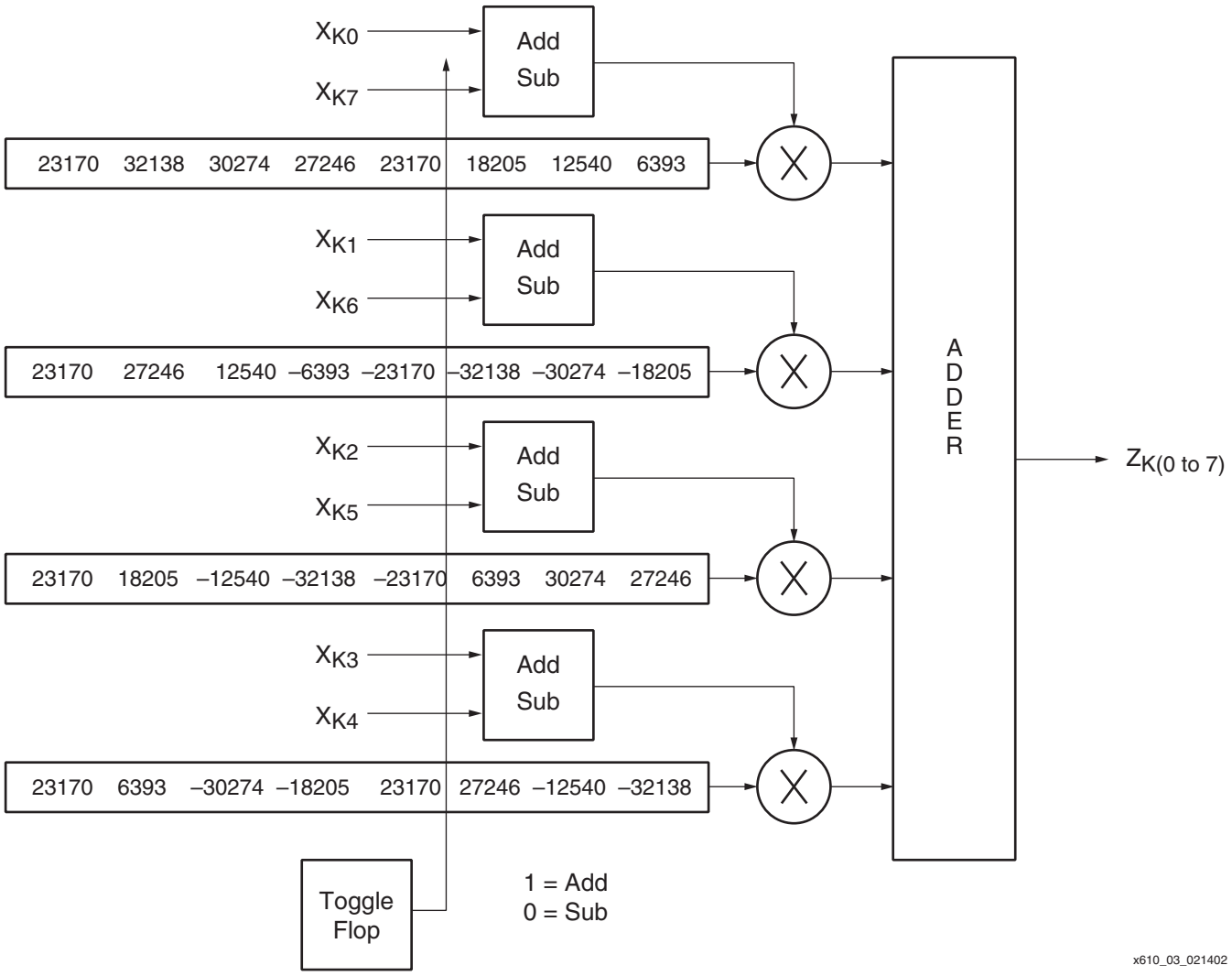

Figure 3: **1D-DCT Implementation (K is 0 - 7)**

The values for $Z_{0(0\ -7)}$ can be calculated in eight clock cycles. (If the adder output is also registered, Z_{07} is calculated at the ninth clock cycle). All 64 values of Z are calculated in 64 clock cycles and then the process is repeated. The values of Z correspond to the 1D-DCT of the input X. Once the Z values are calculated, the 2D-DCT function Y can be calculated from Y = CZ.

$$C = \begin{bmatrix} 23170 & 23170 & 23170 & 23170 & 23170 & 23170 & 23170 & 23170 \\ 32138 & 27246 & 18205 & 6393 & -6393 & -18205 & -27246 & -32138 \\ 30274 & 12540 & -12540 & -30274 & -30274 & -12540 & 12540 & 30274 \\ 27246 & -6393 & -32138 & -18205 & 18205 & 32138 & 6393 & -27246 \\ 23170 & -23170 & -23170 & 23170 & 23170 & -23170 & -23170 & 23170 \\ 18205 & -32138 & 6393 & 27246 & -27246 & -6393 & 32138 & -18205 \\ 12540 & -30274 & 30274 & -12540 & -12540 & 30274 & -30274 & 12540 \\ 6393 & -18205 & 27246 & -32138 & 32138 & -27246 & 18205 & -6393 \end{bmatrix} \quad Z = \begin{bmatrix} z00 & z01 & z02 & z03 & z04 & z05 & z06 & z07 \\ z10 & z11 & z12 & z13 & z14 & z15 & z16 & z17 \\ z20 & z21 & z22 & z23 & z24 & z25 & z26 & z27 \\ z30 & z31 & z32 & z33 & z34 & z35 & z36 & z37 \\ z40 & z41 & z42 & z43 & z44 & z45 & z46 & z47 \\ z50 & z51 & z52 & z53 & z54 & z55 & z56 & z57 \\ z60 & z61 & z62 & z63 & z64 & z65 & z66 & z67 \\ z70 & z71 & z72 & z73 & z74 & z75 & z76 & z77 \end{bmatrix}$$

Each element in the first row in the coefficient matrix C is multiplied by each element in the first column of matrix Z and added together to get the first value Y_{00} of the output matrix Y. To get Y_{01}, each element of row zero in C is multiplied by each element in the first column of Z and is added as before. Multiplying row k of C by column zero of Z results in the coefficient Y_{K0}. All the elements in the first row of matrix Z are multiplied by all the elements in the first column of matrix C.

$$\begin{matrix} 23170\rightarrow \\ 23170\rightarrow \\ 23170\rightarrow \\ 23170\rightarrow \\ 23170\rightarrow \\ 23170\rightarrow \\ 23170\rightarrow \\ 23170\rightarrow \end{matrix} \begin{bmatrix} z00 & z01 & z02 & z03 & z04 & z05 & z06 & z07 \\ z10 & z11 & z12 & z13 & z14 & z15 & z16 & z17 \\ z20 & z21 & z22 & z23 & 24 & z25 & z26 & z27 \\ z30 & z31 & z32 & z33 & z34 & z35 & z36 & z37 \\ z40 & z41 & z42 & z43 & z44 & z45 & z46 & z47 \\ z50 & z51 & z52 & z53 & z54 & z55 & z56 & z57 \\ z60 & z61 & z62 & z63 & z64 & z65 & z66 & z67 \\ z70 & z71 & z72 & z73 & z74 & z75 & z76 & z77 \end{bmatrix}$$

The calculation is implemented by using eight multipliers and storing the coefficients in ROMs. At the first clock, the eight coefficients in row zero of C are multiplied by the eight values in the first column of Z and are added together to result in Y_{00}. At the eighth clock, the eight values of row zero of C are multiplied by the eight values in column eight of Z. The resultant is added to give Y_{07}.

The block diagram for the implementation of the 2 D-DCT is shown in Figure 4

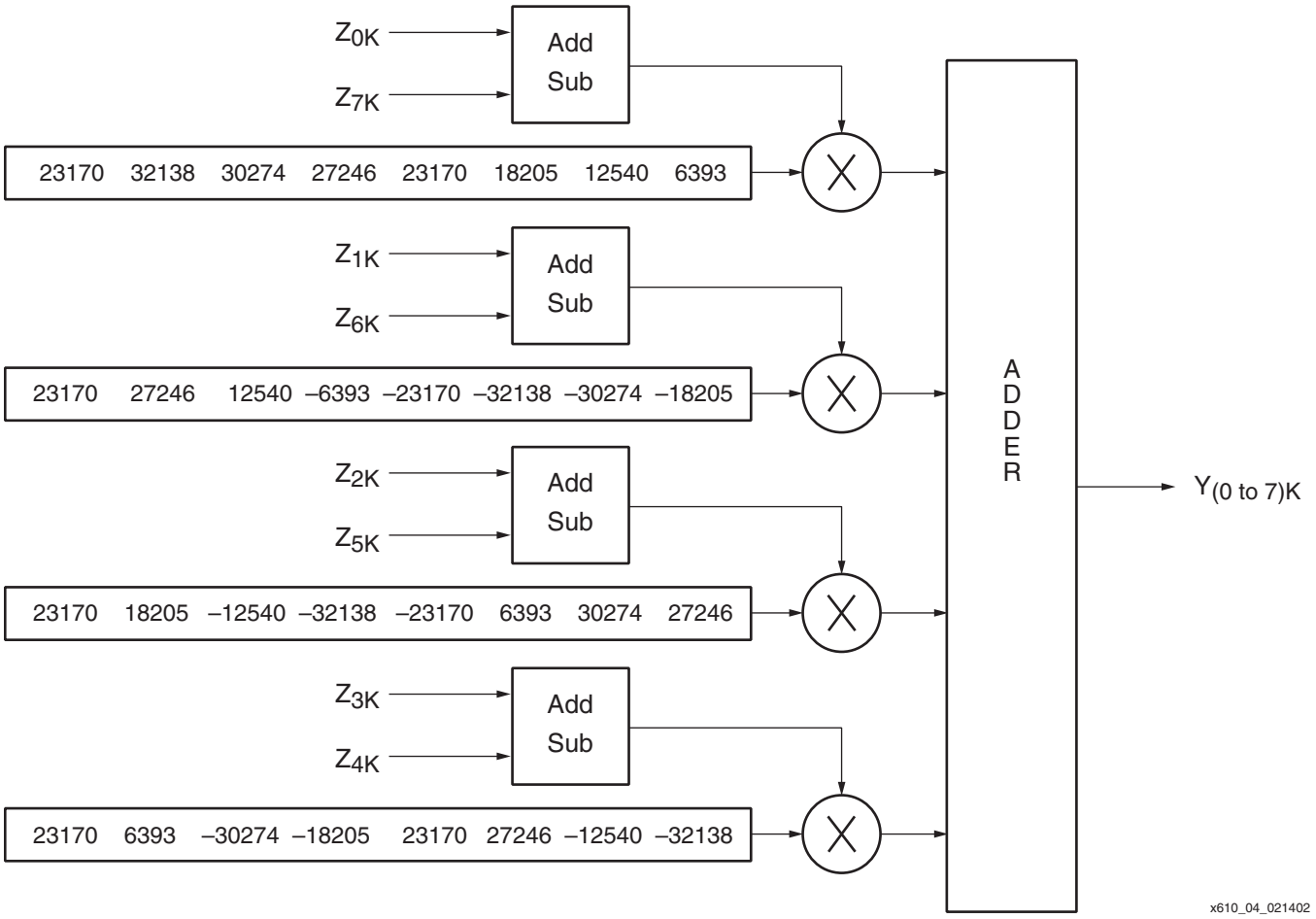

Figure 4: **2D-DCT Implementation (K is 0-7)**

Implementation Description

Preprocessing

In RGB color space, the average value of all the color components is 128. However, in the YCbCr color space the average value of Y is 128, but the value of Cr and Cb is bias zero. The image pixels are preprocessed before going into the DCT coder to provide uniform processing. The preprocessing makes the average value to be zero by subtracting 128 from each pixel value. This value is added back after the inverse DCT operation[1].

Figure 3 and Figure 4 show the block diagrams for reference design implementation. The 1D-DCT values are first calculated and stored in a RAM. The second 1D-DCT is done on the values stored in the RAM. For each 1D implementation, inputs are loaded into an adder/subtractor. The output of the adder/subtractor is fed into a multiplier. The constant coefficient multiplication values are stored in a ROM and fed into the second input of the multiplier. The equations for Z and Y show the even column values are obtained by adding the inputs, and the odd column values are obtained by subtracting the inputs. Thus, for every other clock an addition is done at the inputs. This control is achieved by using a simple toggle flip-flop with the output toggling High or Low to select an adder or a subtractor. The outputs of the four multipliers are added together resulting in the intermediate coefficients. The intermediate coefficients are stored in a RAM. The values stored in the intermediate RAM are read out one column at a time (i.e., every eighth value is read out every clock). This is the input for the second DCT.

Reference Design

Resource Utilization

This application note and reference design includes a basic explanation of DCT. The Verilog and VHDL code shows the implementation of a 2D DCT. The code implements a 2D DCT with 8-bit input data and 9-bit output data. The number of input/output data bits can be changed as needed. The code is available on the Xilinx web site at:
ftp://ftp.xilinx.com/pub/applications/xapp/xapp 610.zip.

The code has been tested and simulated against the vectors given in **"Image and Video Compression Standards," second edition, by Vasudev Bhaskaran and Konstantinos Konstantinides, ISBN 0-7923-9952-8** [1], on page 60. This code has not been tested for IEEE compliance.

Table 1: **Resource Utilization**

Device	Speed Grade	Package	Pre-Map (Synthesis Constraint)	Post-Route	Slices
XCV300E	-8	BG352	92.15 MHz (150 MHz)	64.56 MHz	847(27%)
XC2V250	-6	FG456	182.75 MHz (180 MHz)	64.56 MHz	558 (36%)
XCV300	-6	PQ240	104.62 MHz (100 MHz)	64.56 MHz	848(27%)
XC2S200	-6	FG256	95.00 MHz (80 MHz)	64.56 MHz	853(36%)

References and Recommended Links

1. "Image and Video Compression Standards," second edition, by Vasudev Bhaskaran and Konstantinos Konstantinides, ISBN 0-7923-9952-8
2. A variety of cores developed by Xilinx or by partners are available on the Xilinx web site:
   a. http://www.support.xilinx.com/ipcenter/catalog/logicore/docs/dct_1d.pdf
   b. http://www.xilinx.com/products/logicore/alliance/xentec_spotlight.htm
   c. http://www.support.xilinx.com/ipcenter/dct_lounge/index.htm
3. Application note XAPP208: IDCT Implementation in Virtex Devices for MPEG Applications, v 1.1 (12/99) at: http://www.xilinx.com/xapps/xapp208.pdf
4. White paper WP113: A Spartan-II DCT/IDCT Programmable ASSP Solution at: http://www.xilinx.com/publications/whitepapers/wp_spartan.htm

Conclusion

The DCT design files demonstrate an efficient implementation of the DCT algorithm using Virtex devices. The DCT reference design files can be used to target any Xilinx device. Optimize the code by instantiating the adder/subtractor and multiplier units when targeting Virtex parts. The sample design has an initial latency of 92 clock cycles after which one DCT output is obtained at every clock. Of the 92 clock latency, 64 clocks are used to write in the 64 1D-DCT values into the transpose memory.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
03/06/02	1.0	Initial Xilinx release.
03/20/02	1.1	Updated Reference Design section
04/24/02	1.2	Fixed link to reference design.