# XAPP610 Video Compression using DCT – 8x8 points.

**Summary:** The application note describes a 2 dimensional DCT function implemented on a Xilinx FPGA. Behavioral code is provided for the implementation on any Xilinx device. Some of the advantageous of the module includes parametrizeability and performance guarantee. The code can be further optimized by instantiating embedded adders and multipliers when targeting Virtex2 family. After an initial latency of 92 clock cycles, one DCT value is output at every clock.

## What is compression and why is it needed:

Compression is the process of reducing the size of the data sent and thereby the necessary bandwidth required for the digital representation of a signal. Many applications in video and audio are made possible or more inexpensive because of the ability to compress the signals. Compression technology can result in reduced transmission time because there is less data to be transmitted. It can also decrease the storage requirements, again because there is less data. Signal quality, implementation complexity and communication delay introduced, are potential negative factors that must 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 the same pixel position. Spectral correlation is the correlation of samples of the same sample from multiple sensors.

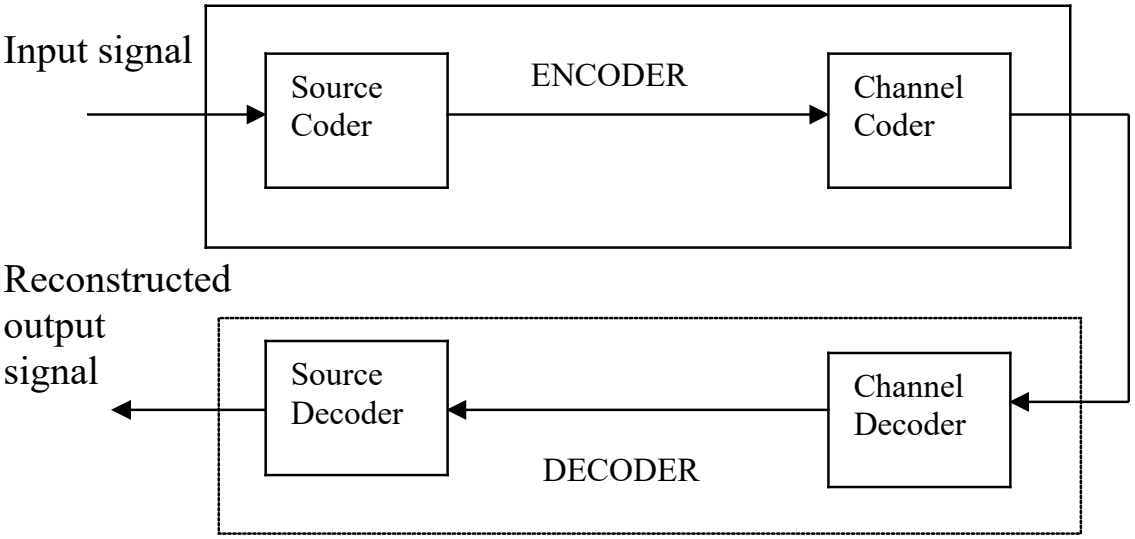


Fig: 1. Block Diagram of a Compression/Decompression System (Image and Video Compression Standards by Vasudev Bhaskaran and Konstantinos Konstantinides)

There are two categories of compression: Lossy compression and lossless compression. In applications like medical systems, image losses can translate into costly medical mistakes; therefore lossless compression methods are used. Luckily, the majority of video and image processing applications do not require the reconstructed data to be

identical to the original data. In these applications, lossy compression schemes can be used, achieving higher compression ratios.

## DCT Compression:

Discrete Cosine Transform (DCT) is a lossy compression scheme in which a NxN image block is transformed from the spatial domain to the DCT domain. DCT decomposes the signal into spatial frequencies 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 for lower frequency coefficients. Because of this , the higher frequency components can be more finely quantized. This is done by the quantization matrix.

Each value in this matrix is pre-scaled by multiplying by a single value, known as the quantizer scale code. This value may range in value from 1-112, and is modifiable on a macroblock basis. Dividing each DCT coefficient with an integer scale factor and rounding the results does quantization. This sets the higher frequency coefficients (in the lower right corner, which are less significant to the compressed picture) to zero by quantizing in larger steps. The low frequency coefficients (in the upper left corner, which 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 $Quantizedvalue_{(i,j)} = \frac{DCT_{(i,j)}}{Qunatizationmatrix_{(i,j)}}$

A sample of the Quantization matrix used for JPEG algorithm is shown below.

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 8x8 block size does not have significant memory requirements and further more a block size greater than 8x8 does not offer significantly better compression. Each picture is divided into 16x16 blocks called a macroblock. Each of the macroblocks is further divided into 16 samples x16 lines. Each

block on which DCT is performed is 8 samples by 8 lines called blocks. Thus each macroblock consists of 4 blocks.

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

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 videoconferencing and videotelephony).

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 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". The 12 bit signed input pixels will provide a 16 bit signed output coefficient for the DCT. (12 bit signed has 11 data bits. For an 8x8 DCT, the 11 data bits can be multiplied 8 times (represented by 3 bits). 11 bits multiplied by 3bits 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

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

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 equation1 into the row part and the column part.

$$C = K \bullet \cos\frac{(2 \cdot col\#+1) \bullet row\# \bullet \pi}{2 \bullet M}$$ , where K = √1/N for row = 0, √2/N for row ≠ 0

$$C^t = K \bullet \cos\frac{(2 \cdot row\#+1) \bullet col\# \bullet \pi}{2 \bullet N}$$ , where K = √1/M for col = 0, √2/M for col ≠ 0

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:

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

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

For the case of 8x8 block region this means that a one dimensional 8 point DCT/IDCT followed by internal double buffer memory followed by another one dimensional 8 point DCT will provide the architecture of the 2D DCT .

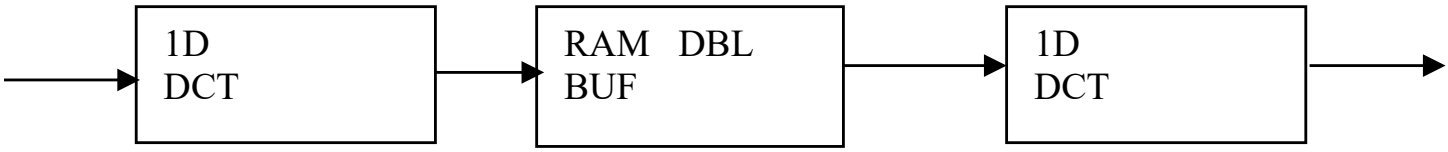


Fig.2. 2D-DCT Using Vector Processing

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

## Behavioral Model for Vector Processing.

Using vector processing, the output Y of an 8x8 DCT for input X is given by

$$Y = C.X.\ C^t,$$

where C is the cosine coefficients and $C^t$ are the transpose coefficients. The above equation can also be written as Y=C.Z, where $Z = X.C^t$. Using the cosine C and inverse cosine $C^t$ numbers, the intermediate value $Z = X.C^t$ can be calculated as follows:

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

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

$$\begin{aligned}
Z_{(0,0)} &= 23170(x_{00}+x_{01}+x_{02}+x_{03}+x_{04}+x_{05}+x_{06}+x_{07}) \\
Z_{(0,1)} &= 32138x_{00}+27246x_{01}+18205x_{02}+6393x_{03}-6393x_{04}-18205x_{05}-27246x_{06}-32138x_{07} \\
&= 32138(x_{00}-x_{07})+27246(x_{01}-x_{06})+18205(x_{02}-x_{05})+6393(x_{03}-x_{04}) \\
Z_{(0,2)} &= 30274(x_{00}+x_{07})+12540(x_{01}+x_{06})-12540(x_{02}+x_{05})-30274(x_{03}+x_{04}) \\
Z_{(0,3)} &= 27246(x_{00}-x_{07})-6393(x_{01}-x_{06})-32138(x_{02}-x_{05})-18205(x_{03}-x_{04}) \\
Z_{(0,4)} &= 23170(x_{00}+x_{07})-23170(x_{01}+x_{06})-23170(x_{02}+x_{05})+23170(x_{03}+x_{04}) \\
Z_{(0,5)} &= 18205(x_{00}-x_{07})-32138(x_{01}-x_{06})+6393(x_{02}-x_{05})+27246(x_{03}-x_{04}) \\
Z_{(0,6)} &= 12540(x_{00}+x_{07})-30274(x_{01}+x_{06})+30274(x_{02}+x_{05})-12540(x_{03}+x_{04}) \\
Z_{(0,7)} &= 6393(x_{00}-x_{07})-18205(x_{01}-x_{06})+27246(x_{02}-x_{05})-32138(x_{03}-x_{04})
\end{aligned}$$

Or

$$\begin{aligned}
Z_{(k,0)} &= 23170(x_{k0}+x_{k1}+x_{k2}+x_{k3}+x_{k4}+x_{k5}+x_{k6}+x_{k7}) \\
Z_{(k,1)} &= 32138(x_{k0}-x_{k7})+27246(x_{k1}-x_{k6})+18205(x_{k2}-x_{k5})+6393(x_{k3}-x_{k4}) \\
Z_{(k,2)} &= 30274(x_{k0}+x_{k7})+12540(x_{k1}+x_{k6})-12540(x_{k2}+x_{k5})-30274(x_{k3}+x_{k4)} \\
Z_{(k,3)} &= 27246(x_{k0}-x_{k7})-6393(x_{k1}-x_{k6})-32138(x_{k2}-x_{k5})-18205(x_{k3}-x_{k4}) \\
Z_{(k,4)} &= 23170(x_{k0}+x_{k7})-23170(x_{k1}+x_{k6})-23170(x_{k2}+x_{k5})+23170(x_{k3}+x_{k4}) \\
Z_{(k,5)} &= 18205(x_{k0}-x_{k7})-32138(x_{k1}-x_{k6})+6393(x_{k2}-x_{k5})+27246(x_{k3}-x_{k4}) \\
Z_{(k,6)} &= 12540(x_{k0}+x_{k7})-30274(x_{k1}+x_{k6})+30274(x_{k2}+x_{k5})-12540(x_{k3}+x_{k4}) \\
Z_{(k,7)} &= 6393(x_{k0}-x_{k7})-18205(x_{k1}-x_{k6})+27246(x_{k2}-x_{k5})-32138(x_{k3}-x_{k4})
\end{aligned}$$

where k = 0,2,….,7.

Each element in the first row in 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 0 in X is multiplied by each element in column1 of $C^t$ and added and so on. It can be seen that 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.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| *x00* → | 23170 | 32138 | 30274 | 27246 | 23170 | 18205 | 12540 | 6393 |
| *x01* → | 23170 | 27246 | 12540 | -6393 | -23170 | -32138 | -30274 | -18205 |
| *x02* → | 23170 | 18205 | -12540 | -32138 | -23170 | 6393 | 30274 | 27246 |
| *x03* → | 23170 | 6393 | -30274 | -18205 | 23170 | 27246 | -12540 | -32138 |
| *x04* → | 23170 | -6393 | -30274 | 18205 | 23170 | -27246 | -12540 | 32138 |
| *x05* → | 23170 | -18205 | -12540 | 32138 | -23170 | -6393 | 30274 | -27246 |
| *x06* → | 23170 | -27246 | 12540 | 6393 | -23170 | 32138 | -30274 | 18205 |
| *x07* → | 23170 | -32138 | 30274 | -27246 | 23170 | -18205 | 12540 | -6393 |

The above can be implemented 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 to give eight products ($_{P00\_0}$ to $_{P00\_7}$). At the eighth clock, the eight inputs are multiplied by the eight values in column eight to give 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 as

$$Z_{00} = P_{00\_0} + P_{00\_1} + P_{00\_2} + P_{00\_3} + P_{00\_4} + P_{00\_5} + P_{00\_6} + P_{00\_7}$$
$$Z_{01} = P_{01\_0} + P_{01\_1} + P_{01\_2} + P_{01\_3} + P_{01\_4} + P_{01\_5} + P_{01\_6} + P01\_7$$
$$Z_{ij} = Pij\_0 + P_{ij\_1} + P_{ij\_2} + P_{ij\_3} + P_{ij\_4} + P_{ij\_5} + P_{ij\_6} + P_{ij\_7}$$

Where i = 0to7, matrix X row number and j = 0to7 matrix $C^t$ column number.

The intermediate values for the second row of Z, Z1K, are computed by using P1K values. The 8x8 Z matrix can be calculated using the above equations.

The block diagram for the implementation of the 1D-DCT as described above is as shown in Fig. 3.

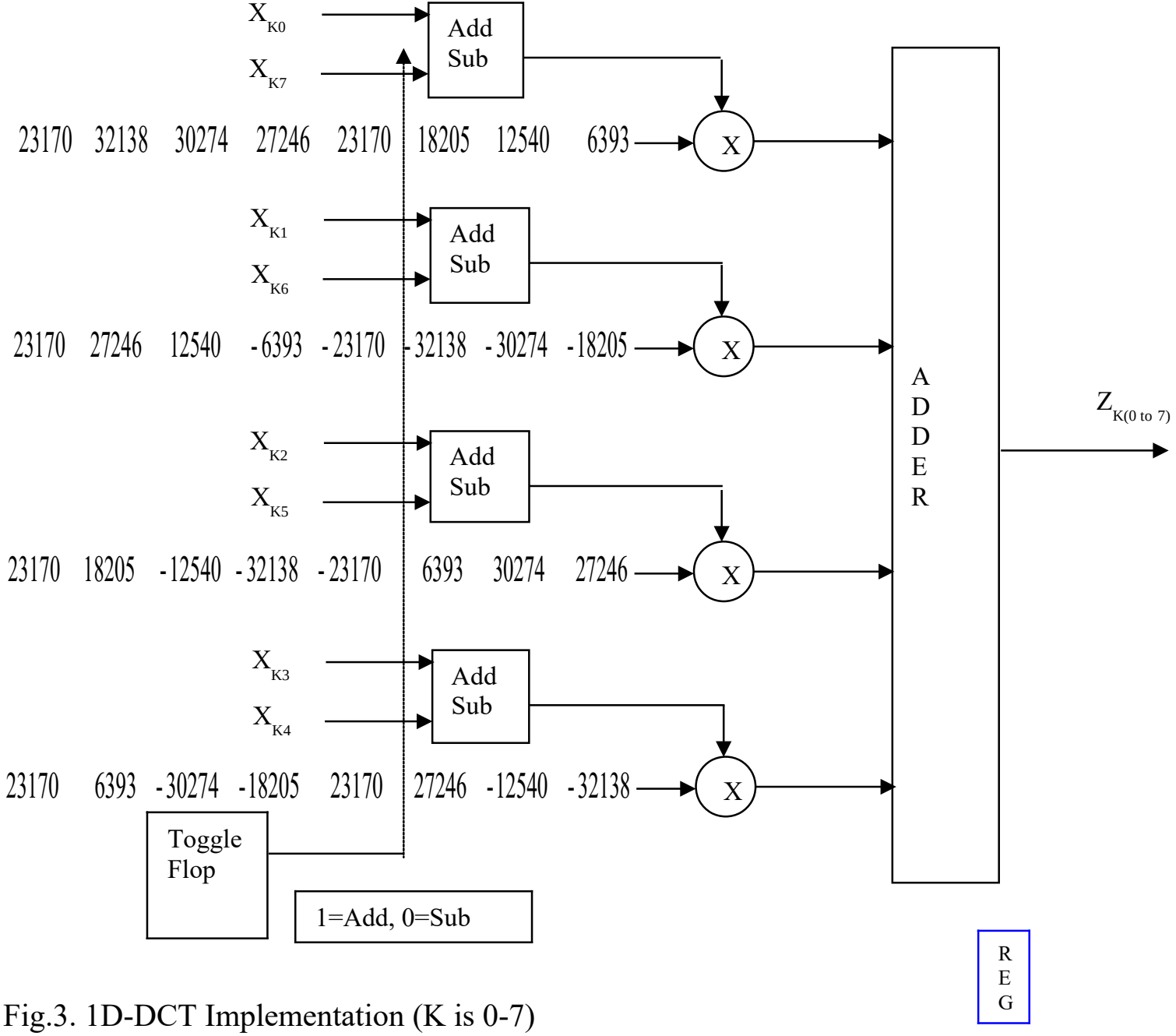


Fig.3. 1D-DCT Implementation (K is 0-7)

The values for $Z_{0(0-7)}$ can be calculated in 8 clock cycles. (If the adder output is also registered, $Z_{07}$ is calculated at the 9th clock cycle.) All the 64 values of Z are calculated in 64 clock cycles and the process then repeats. 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

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

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

$$Y = CZ$$
$$Y_{0x} = 23170(Z_{0x} + Z_{7x}) + 23170(Z_{1x} + Z_{6x}) + 23170(Z_{2x} + Z_{5x}) + 23170(Z_{3x} + Z_{4X})$$
$$Y_{1X} = 32138(Z_{0x} - Z_{7x}) + 27246(Z_{1x} - Z_{6x}) + 18205(Z_{2x} - Z_{5x}) + 6393(Z_{3x} - Z_{4X})$$
$$Y_{2x} = 30274(Z_{0x} + Z_{7x}) + 12540(Z_{1x} + Z_{6x}) - 12540(Z_{2x} + Z_{5x}) - 30274(Z_{3x} + Z_{4X})$$
$$Y_{3X} = 27246(Z_{0x} - Z_{7x}) - 6393(Z_{1x} - Z_{6x}) - 32138(Z_{2x} - Z_{5x}) - 18205(Z_{3x} - Z_{4X})$$
$$Y_{4x} = 23170(Z_{0x} + Z_{7x}) - 23170(Z_{1x} + Z_{6x}) - 23170(Z_{2x} + Z_{5x}) + 23170(Z_{3x} + Z_{4X})$$
$$Y_{5X} = 18205(Z_{0x} - Z_{7x}) - 32138(Z_{1x} - Z_{6x}) + 6393(Z_{2x} - Z_{5x}) + 27246(Z_{3x} - Z_{4X})$$
$$Y_{6x} = 12540(Z_{0x} + Z_{7x}) - 30274(Z_{1x} + Z_{6x}) + 30274(Z_{2x} + Z_{5x}) - 12540(Z_{3x} + Z_{4X})$$
$$Y_{7X} = 6393(Z_{0x} - Z_{7x}) - 18205(Z_{1x} - Z_{6x}) + 27246(Z_{2x} - Z_{5x}) - 32138(Z_{3x} - Z_{4X})$$

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 0 in C is multiplied by each element in column1 of Z and added and so on. It can be seen that when row k of C is multiplied by column 0 of Z, we get 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.

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

The above can be implemented using eight multipliers and storing the coefficients in ROMs. At the first clock, the eight coefficients in row 0 of C are multiplied by the eight values in column 1 of Z and are added together to give $Y_{00}$. At the eighth clock, the eight values of row 0 of C are multiplied by the eight values in column eight of Z, which are added to give $Y_{07}$.

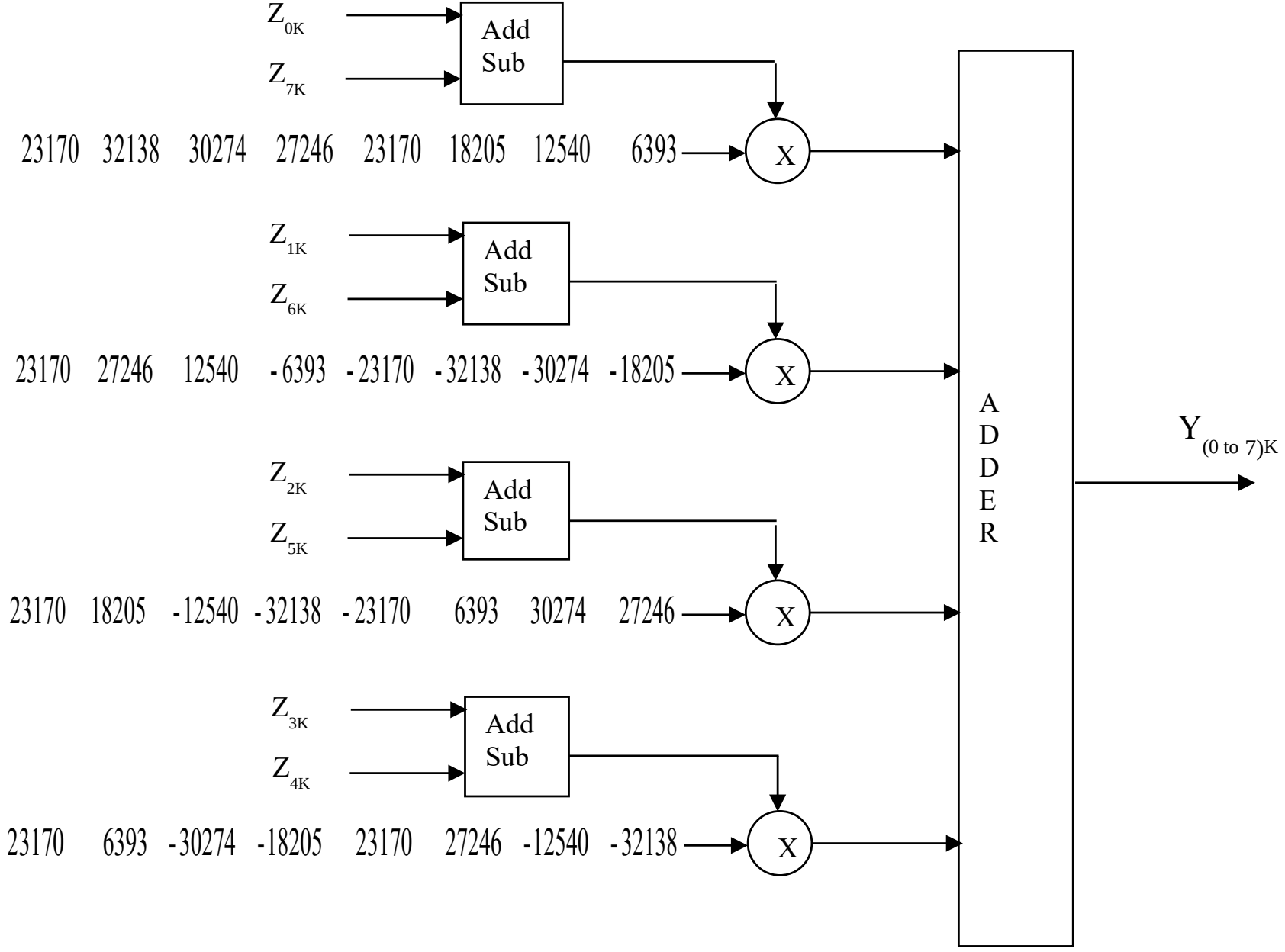


Fig.4. 2D-DCT Implementation (K is 0-7)

## Implementation Description:

**Preprocessing**: In RGB color space, the average value of all the color components are 128. In YCbCr color space however, the average value of Y is 128 but that of Cr and Cb bias zero. The image pixels are preprocessed before going into the DCT coder so that uniform processing is provided. The preprocessing makes the average value to be zero by subtracting 128 from each pixel value. This value is adder back after the inverse DCT operation.

The schematic for the implementation are given in Fig3, 4. The 1D-DCT values are first calculated and stored in a RAM. The 2nd 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 2nd input of the multiplier. From the equations for Z and Y, it can be seen that 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 can be achieved by using a simple toggle flop with the output toggling to a 1 or 0 to select an adder or a subtractor. The outputs of the 4 multipliers are added together to give the intermediate coefficients which are stored in a RAM. The values stored in the intermediate RAM are read out one column at a time (i.e., every 8th value is read out every clock) and this is the input for the 2nd DCT.

## Resource Utilization:

| Device | Pre-map(Synthesis constraint) | Post-route | Slices |
|---|---|---|---|
| V300E,BG352-8 | 92.15MHz(150MHz) | 64.56MHz | 847(27%) |
| 2V250,FG456-6 | 182.75MHz (180 MHz) | 140.35MHz | 558(36%) |
| XCV300,PQ240-6 | 104.62MHz(100MHz) | 51.98MHz | 848(27%) |
| XC2S200,FG256-6 | 95.00MHz(80MHz) | 55.27MHz | 853(36%) |
| (Synplify Pro Verilog + Foundation 4.1.03I) | | | |

## Conclusion:

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

## DCT/IDCT Solutions Available at Xilinx:

Xilinx DCT Lounge
CAST Forward DCT
CAST DCT/IDCT
***Coregen***

***Logicore by Amphion***

Product Features
- Combined DCT/IDCT Core
- Supports Virtex-II, Virtex, Virtex-E and Spartan-II devices
- Continuous one symbol per cycle processing capability
- Internal precision:
  - 14 bit cosine coefficients
  - 15 bit transpose memory
- Optimized for specific Xilinx architecture
- Fully compliant with the JPEG standard (ISO/IEC 10918-1)

http://www.support.xilinx.com/ipcenter/dct_lounge/index.htm
Amphion Forward DCT **
Amphion IDCT **

***Alliance core by TiLAB***
TILAB DCT/IDCT

## Reference Documents and Designs:

Additional DCT-IDCT materials can be found in
http://www.support.xilinx.com/ipcenter/catalog/logicore/docs/dct_1d.pdf
http://stagelinx.xilinx.com:80/products/logicore/alliance/xentec_spotlight.htm
XAPP208 IDCT Implementation in Virtex Devices for MPEG Applications, v 1.1 (12/99)
WP113: A Spartan-II DCT/IDCT Programmable ASSP Solution

## References:


1. Image and Video Compression Standards ,Second Edition, by Vasudev Bhaskaran and Konstantinos Konstantinides , ISBN 0-7923-9952-8
2. Bob Turney, Senior Staff R&D Engineer, Multimedia, Video & Image Processing, Xilinx Labs, Xilinx Inc. turney@xilinx.com


Replacement:

www.ocean-logic.com/pub/OL_DCT.pdf