WP113 (v1.0) February 25, 2000

A Spartan-II DCT/IDCT Programmable ASSP Solution

Author: Antolin Agatep

Overview

This paper presents an overview of Discrete Cosine Transform (DCT) and Inverse Discrete Cosine Transform (IDCT) solutions using XIlinx Spartan™-II components with IP core technology from Xilinx AllianceCORE™ partner Xentec, Inc.

Introduction

XIlinx Spartan-II and AllianceCORE: The Ideal DCT/IDCT Solution

The Spartan-II family of FPGAs is effectively taking over many traditional ASSP markets, primarily because of its ability to keep pace with the evolution of new, highly efficient IP core algorithms. Traditional ASSP solutions cannot do this without incurring expensive overhead costs due to re-spins.

The Spartan-II family of FPGAs are performance and price competitive solutions. The fast pace of twenty-first century computing demands the very best solutions that target aggressive time to market windows of opportunities. From deeply embedded solutions to networking solutions, Xilinx FPGAs provide unparalleled benchmarks of flexibility, productivity and creativity.

Through Xentec, Inc., on the Web at http://www.xentec-inc.com, Xilinx is offering a novel DCT/IDCT solution that is both compact and has high-performance. Xentec's core provides an effective combination for systems that require fast and efficient computation of DCT and IDCT. Xentec is a new member of Xilinx AllianceCORE, which comprises of companies providing high quality, high performance, synthesizable core solutions targeting the Spartan-II and other Xilinx FPGA families.

Data Storage and Compression with DCT/IDCT

The area of compression is a vast subject, but the focus of this White Paper is the discrete cosine transform (DCT) and its inverse, known as the inverse discrete cosine transform (IDCT). DCT is actually the "even" part of the Fourier series, meaning only the part of the infinite series that contains cosines.

While the idea of unlimited data storage is certainly pleasant to imagine, practical design demands innovative solutions to the limitations imposed by real-world storage devices. In the case of a picture taken by a digital camera, the information is stored in nonvolatile FLASH memory. The goal is to compress the picture's information as much as practicable so that a larger number pictures can be taken and stored at one time.

In the case of multimedia systems, video and audio information can be efficiently transmitted from point to point using compression. Compression allows more information to be squeezed into the video and audio transmission medium. This means higher video frame rates, better audio quality, additional system features like interactivity, and other extra tasks the system can perform due to the extra time and space compression allows. Basically, there is extra time because compression generates less video and audio information. The microprocessor is effectively free during the saved time to attend to other tasks.

The compression occurs when a specific value or "threshold" is applied. This technique eliminates those values of the calculated DCT which do not rise above the set threshold. The end result is a reduced number of bits in the 1-D data sequence.

To visualize thresholding, think of a picture and then imagine dimming the light incrementally. By increasing the "threshold value" in this analogy, the light eventually becomes too dim for the picture to be seen. Thresholding in 2-D is generally applied to black and white images.

DCT/IDCT Applications Examples

Some typical DCT/IDCT applications

- Image storage
- Data compression
- Video compression
- Dolby AC2 and AC3, 1-D DCT
- JPEG (still images), 2-D DCT spatial compression
- MPEG1 and MPEG2, 2-D DCT plus motion compensation
- Cable TV networks use MPEG2 as the standard for compressing and decompressing video for distribution and broadcasting.
- DBS (Direct Broadcast Satellite) will use MPEG2 for direct broadcast.
- HDTV (High-Definition Television also known as ATV)
- DVD uses MPEG2 for the encoding which helps make VOD (video on demand) possible.
- Medical imaging
- Image and pattern recognition
- Biomedical signals like EEGs and ECGs
- Speech information compression

JPEG and MPEG Encoding / Decoding

Figure 1 shows an example of where DCT and IDCT are being applied in a real-world application, a JPEG encoder and a JPEG decoder. JPEG is most often used in digital still image applications like digital cameras. JPEG can also be used in video, but it doesn't take advantage of redundant information in the frames. MPEG, another standard which uses DCT, handles real-time video more efficiently through noting the locations of redundant information within video frames and discarding it. This greatly improves efficiency, compute time and transmission bandwidth.

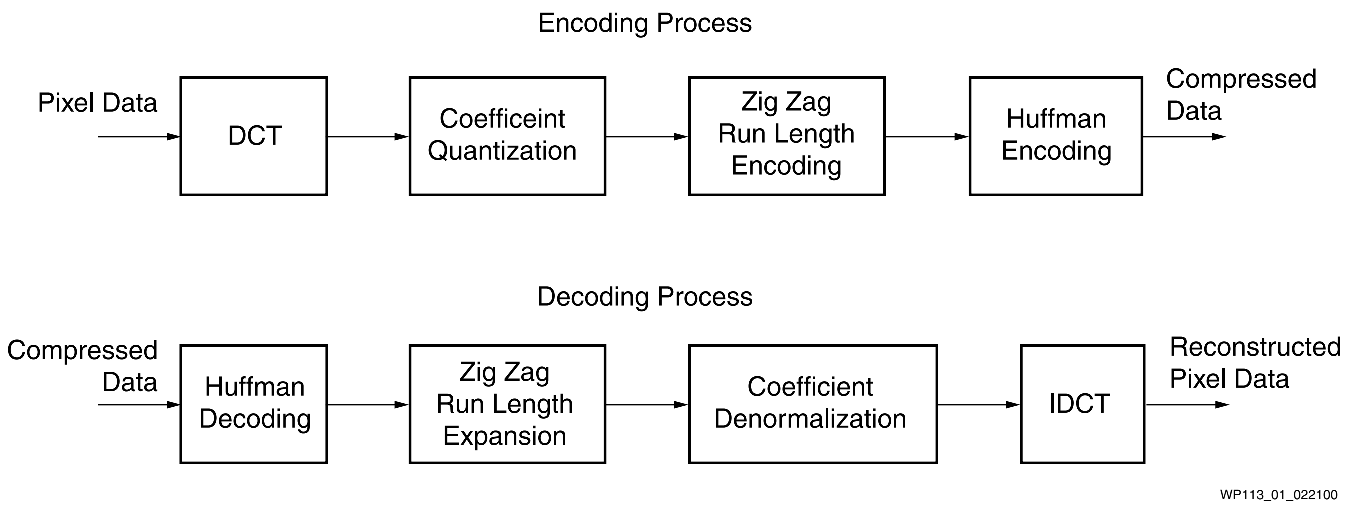

Figure 1: **JPEG Encoding and Decoding**

Figure 2 depicts the MPEG-2 compression, or encoding, process block diagram. The two main steps to MPEG-2 encoding are spatial compression within each frame using block encoding techniques, and temporal compression between nearby frames (anchor frames) within the video stream which removes redundant frame-to-frame information. This second step, which is called motion estimation, is not used on all the compressed frames.

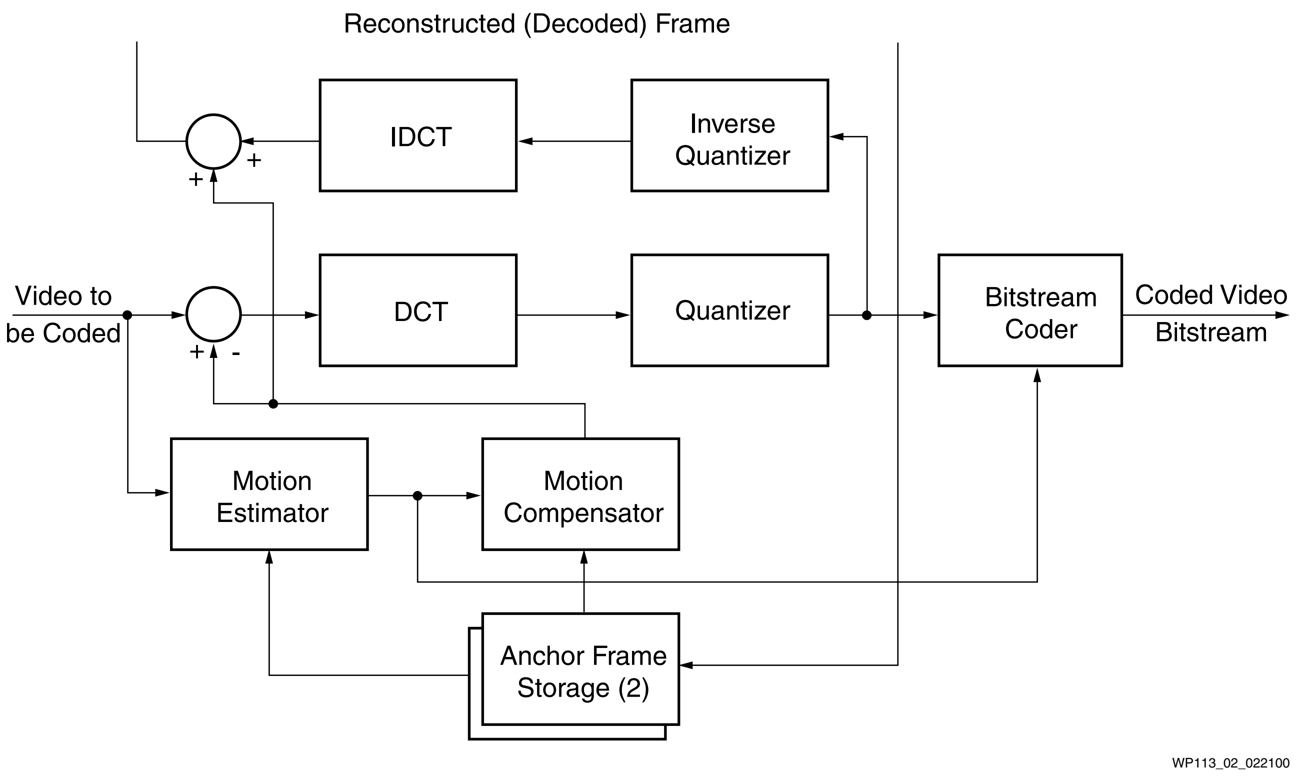

Figure 2: **MPEG Encoder Block Diagram** (courtesy: C-Cube Microsystems)

MPEG-2 has become synonymous with DVD primarily because DVD has adopted MPEG-2 as its main means of dealing with motion video. The MPEG-2 standard delivers four times more information than the regular MPEG-1 standard, thereby delivering the improved video quality everyone expects from DVD technology.

The following figures, Figure 3 and Figure 4, depict diagrams of DVD mastering and DVD player architecture, respectively. It can be observed within these diagrams where MPEG-2 and AC-3 are implemented, which use DCT and IDCT. Also note that the DSP and microcontroller blocks can be easily replaced by available Xilinx synthesizable core solutions as well.

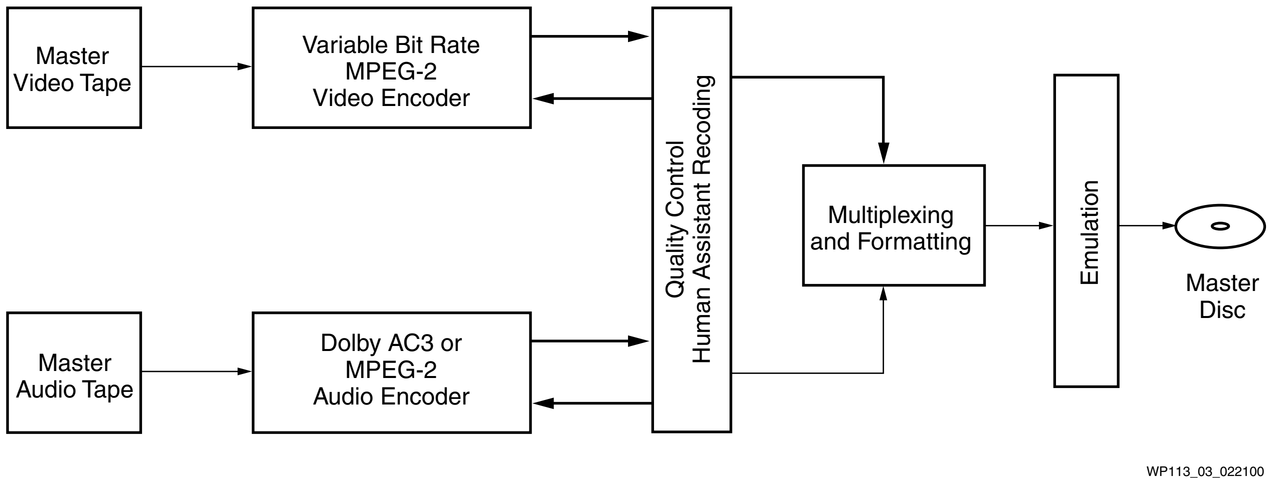
Master Video Tape
Variable Bit Rate MPEG-2 Video Encoder
Master Audio Tape
Dolby AC3 or MPEG-2 Audio Encoder
Quality Control Human Assistant Recoding
Multiplexing and Formatting
Emulation
Master Disc
WP113_03_022100

Figure 3: **DVD Mastering Diagram** (courtesy: C-Cube Microsystems)

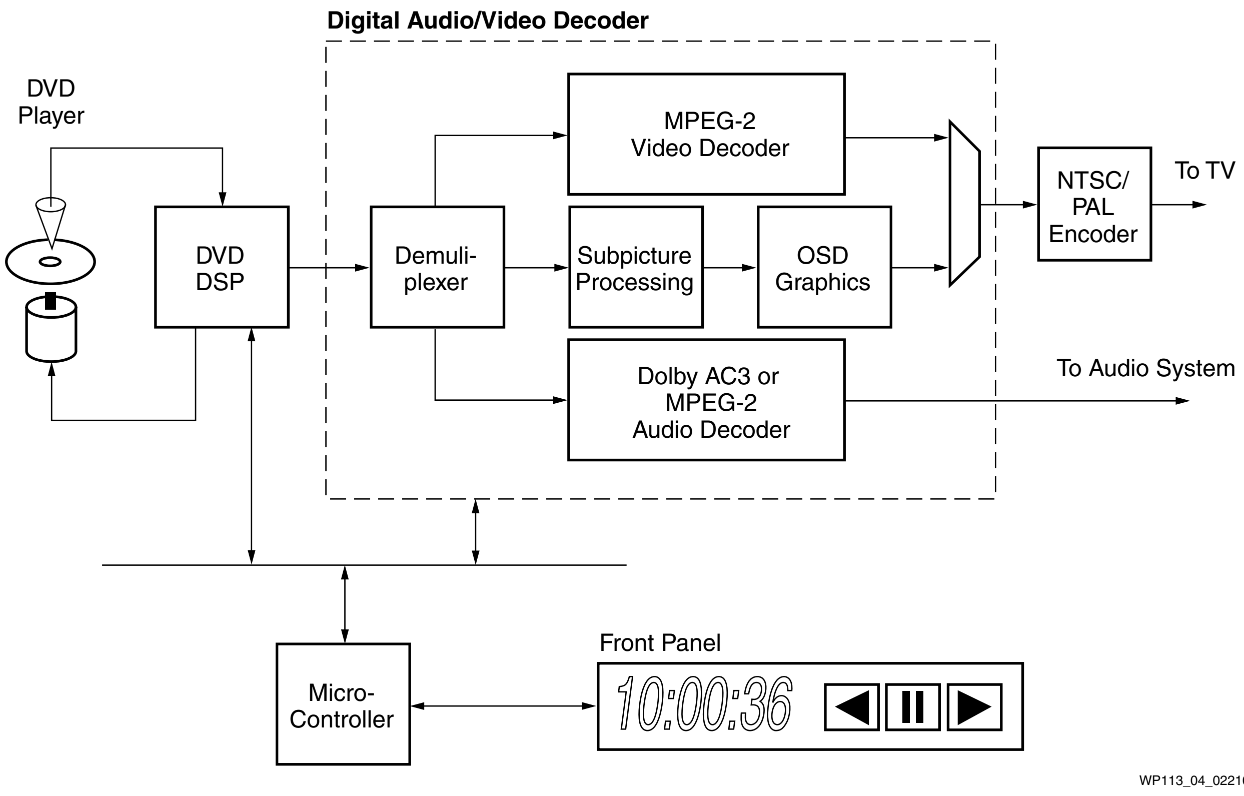

Figure 4: **DVD Player Architecture** (courtesy: C-Cube Microsystems)

How it Works

Mathematical Computation for a One-Dimensional DCT

One-dimensional DCT is commonly used on a sequence of digital information like voice or heartbeat information in an ECG. Two-dimensional DCT is applied generally to data sets which have a naturally two-dimensional characteristic, like a digital image in the form of a picture taken by a digital camera.

Calculating a two-dimensional DCT by hand, which is what's used in compressing a digital image for example, is very tedious; therefore, a one-dimensional computation[1] is presented here.

In a data compression system, for example, the data is transformed first and then the newly calculated values are threshold limited to a magnitude of 0.375. Assuming the data sequence to be {1, 2, 0, 5}, the one-dimensional Discrete Cosine Transform formula

$$X_c(k) = (1/\mathrm{N}) \sum_{n=0}^{N-1} x_n \cos(k2\pi n/\mathrm{N}),\ k = 0, 1, 2, ..., \mathrm{N}\text{-}1$$

will be used.

Let $x_0 = 1$, $x_1 = 2$, $x_2 = 0$, $x_3 = 5$:

$$X_c(0) = (¼)x_0 \cos 0 + x_1 \cos 0 + x_3 \cos 0 = (¼)(1 + 2 + 0 + 5) = 2$$

$$\begin{aligned} X_c(1) &= (¼) \sum_{n=0}^{3} x_n \cos(2\pi n/4) \\ &= (¼) \sum_{n=0}^{3} x_n \cos(n\pi/2) \\ &= (¼)[x_0 + x_1 \cos(\pi/2) + x_2 \cos(2\pi/2) + x_3 \cos(3\pi/2)] \\ &= (¼)[1 + 0 + 0 + 0] = 0.25 \\ X_c(2) &= (¼)(x_0 + x_1 + x_2 - x_3) = (¼)(1 - 2 + 0 - 5) = -6 \\ X_c(3) &= (¼)(1 + 0 + 0 + 0) = 0.25 \end{aligned}$$

Therefore, the entire DCT sequence is {2, 0.25, -6, 0.25}. The values that remain after thresholding (|values| > 0.375) are 2 and –6.

It is evident that two of the 0.25 values are lost in the original sequence—and in fact, this is what's happening with the analogy of dimming the light. The outstanding result is that there is a lesser number of values, specifically 50% less in this case! When the picture is too dark to see, it is the point when the threshold value is high enough that all or the majority of the data in the transformed sequence is rejected.

The one dimensional Inverse Discrete Cosine Transform (IDCT) is given by

$$X_c(k) = \sum_{n=0}^{N-1} c[u]\, x_n \cos(k2\pi n/\mathrm{N}),\ k = 0, 1, 2, ..., \mathrm{N}\text{-}1$$

where:

X_n is the inverse DCT result

$$c[u] = \begin{cases} 1, \text{ for } u = 0, \\ 2, \text{ for } u = 1, 2, 3, ..., N\text{-}1 \end{cases}$$

The fact that the Discrete Cosine Transform gives a mean-squared error result that is close to the theoretical limit of the Karhuenen-Loeve transform is the reason why the DCT is very prevalent in two-dimensional image compression.

"Mean-squared error" is basically the summation over the area (2-D data set) of the square of the difference between the individual elements of the original and recovered image divided by the square of the width of the image.

The definition[2] of the DCT for an N by N image is

$$F[u, v] = 1/N^2 \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} f[m, n] \cos[(2m + 1)u\pi/2N] \cos[(2n + 1)v\pi/2N]$$

where:

u, v = discrete frequency variables (0, 1, 2, …, N - 1)

f[m, n] = N by N image pixels (0, 1, 2, …, N - 1)

F[u, v] = the DCT result

The inverse DCT (IDCT) definition is

$$f[m, n] = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} c[u] c[v] F[u, v] \cos[(2m + 1)u\pi/2N] \cos[(2n + 1)v\pi/2N]$$

where:

m, n = image result pixel indices (0, 1, 2, …, N – 1)

F[u, v] = N by N DCT result

$$c[\lambda] = \begin{cases} 1, \text{ for } \lambda = 0 \\ 2, \text{ for } \lambda = 1, 2, 3, ..., N\text{-}1 \end{cases}$$

f[m, n] = N by N IDCT result

DCT/IDCT Process Flow

The general flow of transforming a sampled image is shown by Figure 5 and the reverse of this, which is the recovery of the image from the transformed or rather compressed image information is shown by Figure 6. Also, note that the Coefficient Selection block of Figure 5 is actually the thresholding part of the previous 1-D DCT example calculation.

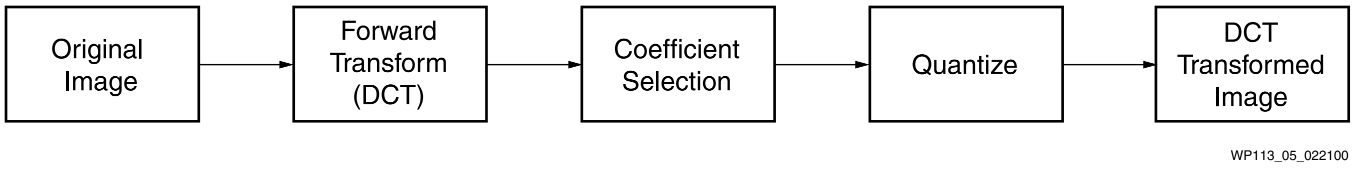

Figure 5: **DCT Used in Compression**

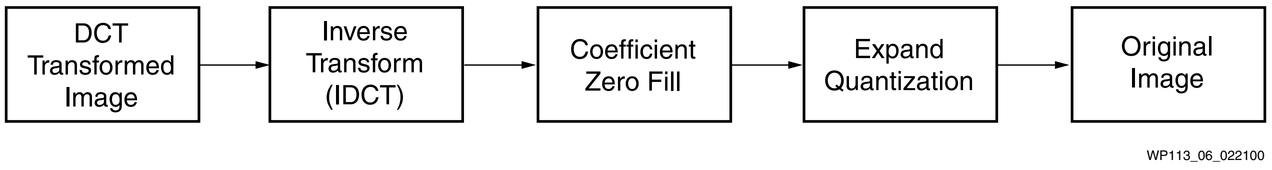

Figure 6: **IDCT Used in Recovery of Image from Compressed Information**

In Figure 7, note that the recovered image is not equal to the original image but rather a good approximation to it. The transformation process in which the DCT is used gets rid of certain values. This transformation process using the DCT can be done to a picture because it contains a lot of redundant information.

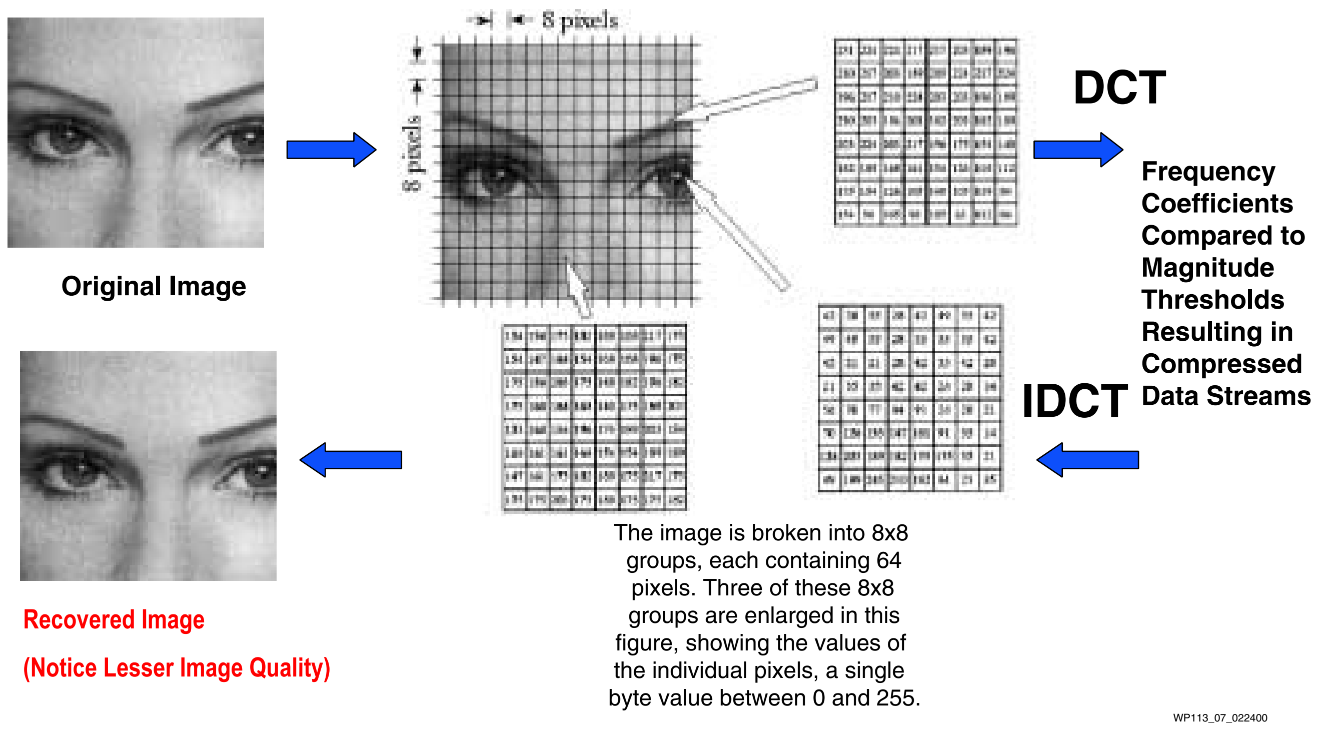
Original Image
8 pixels
8 pixels
DCT
Frequency Coefficients Compared to Magnitude Thresholds Resulting in Compressed Data Streams
IDCT
Recovered Image
(Notice Lesser Image Quality)
The image is broken into 8x8 groups, each containing 64 pixels. Three of these 8x8 groups are enlarged in this figure, showing the values of the individual pixels, a single byte value between 0 and 255.
WP113_07_022400

Figure 7: **Compressing an Image Using DCT/IDCT**
(courtesy: The Scientist and Engineer's Guide to Digital Signal Processing by Steven W. Smith)

Image Block Breakdown

Figure 8 shows[4] that the typical picture frame, whether that of a still image or that of a video, is divided into smaller more manageable blocks which are easily fed into the DCT transform block. This is also true for the reverse process in which the IDCT transform is used.

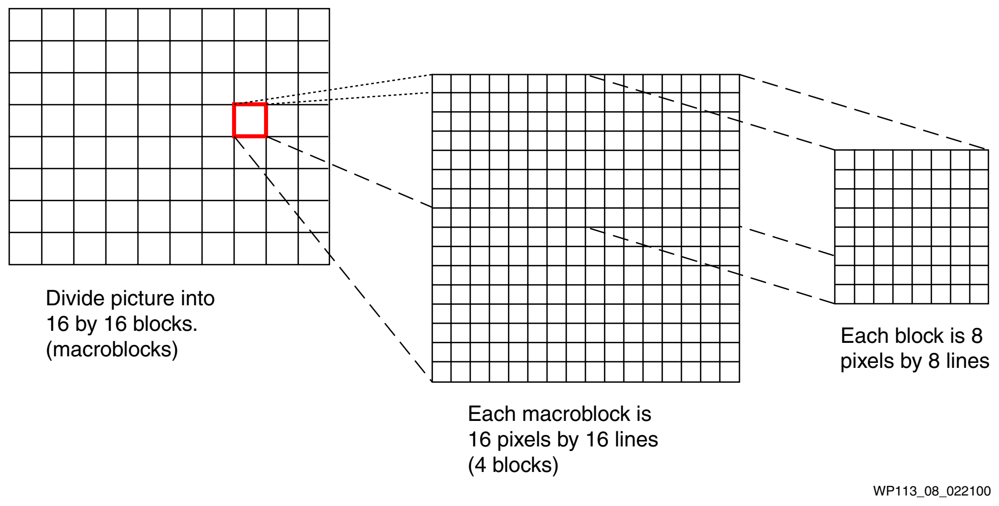

Figure 8: **Picture Breakdown into Macroblocks, Blocks and Pixels**

Specifically, Figure 8 breaks down the regular image into 16 x 16 blocks called macroblocks and then these macroblocks are further subdivided into sixteen by sixteen parts of which four 8 x 8 blocks can be DCT transformed readily, depicted in Figure 9, which shows the conceptual transformation of an eight by eight block. This process takes the 8 x 8 block from the time domain to the frequency domain.

The reason in applying the DCT transform to an image is to bring out a set of numbers, also referred to as coefficients, shown in Figure 9. These coefficients represent the image but are uncorrelated, which means that each number in the 2-D array that represents the image is unique. A coefficient's usefulness is determined by its variance over a set of images, as in video's case. If a coefficient maintains a value over a set of images, then it does not convey a lot of information about the difference in the images in this set and it can be replaced by a constant in the receiving end without affecting the quality. But if a coefficient has a lot of variance over a set, then it cannot be removed without affecting the picture quality.

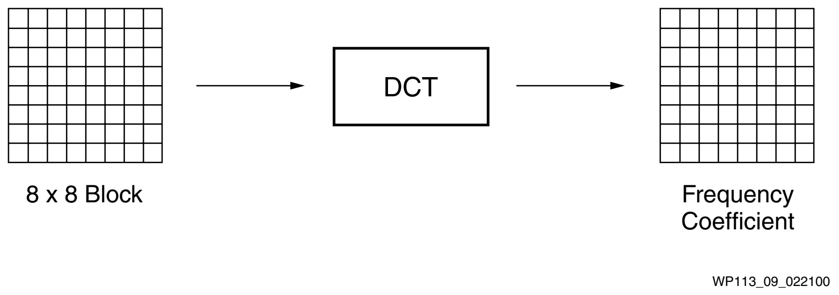

Figure 9: **DCT Transformation of an 8 x 8 Block**

The 8 x 8 block is generally used in DCT computation because experiments have shown little compaction gains can be achieved by using larger transform sizes, especially in light of the increased implementation complexity. A fast DCT algorithm will require approximately twice the number of arithmetic operations per sample when the linear transform point size is doubled.

The best compaction efficiency has been demonstrated using locally adaptive block sizes (e.g., 16x16, 16x8, 8x8, 8x4, and 4x4).[6]

Function Plots

Figure 10 illustrates the contour plot of the function Cos(x) * Cos(y).

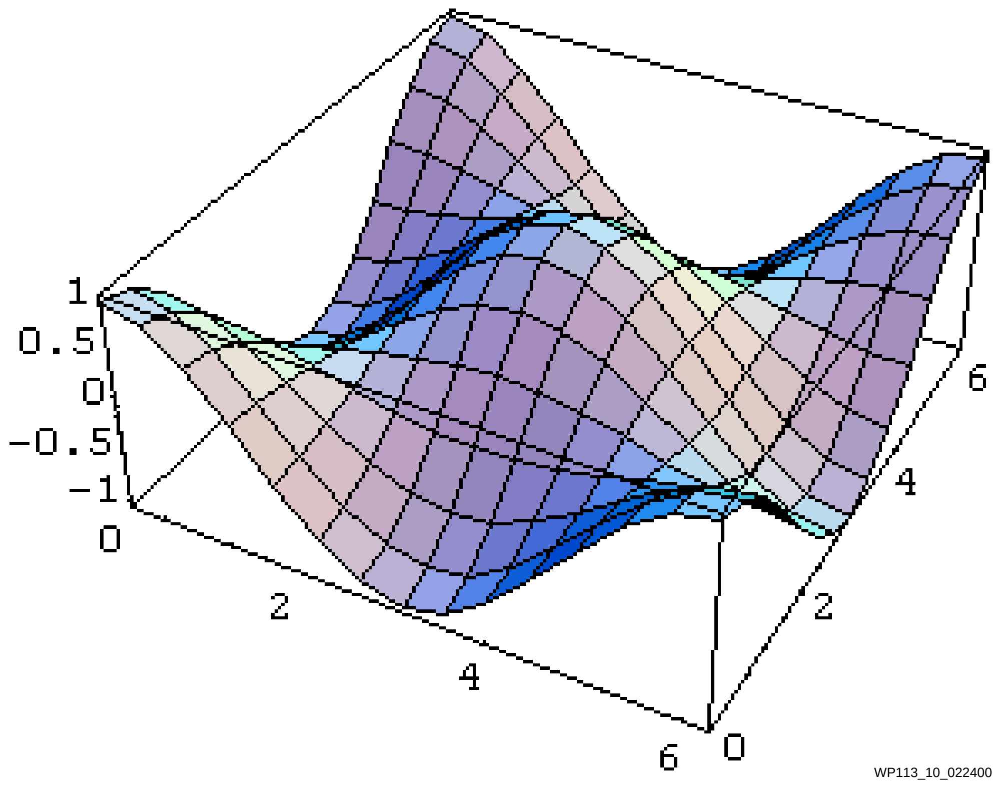

Figure 10: **Plot of f(x,y) = Cos(x) * Cos(y) for Values of x and y from 0 to 2π**

The DCT and IDCT equations do use this function, and the interesting part of the generated surface is that the minimum value can be readily checked to be zero, and the maximum value is one. This is expected, because the product of the two cosines produces an absolute value effect.

Figure 11 below shows the top view of the Cos(x) ∗ Cos(y) function.

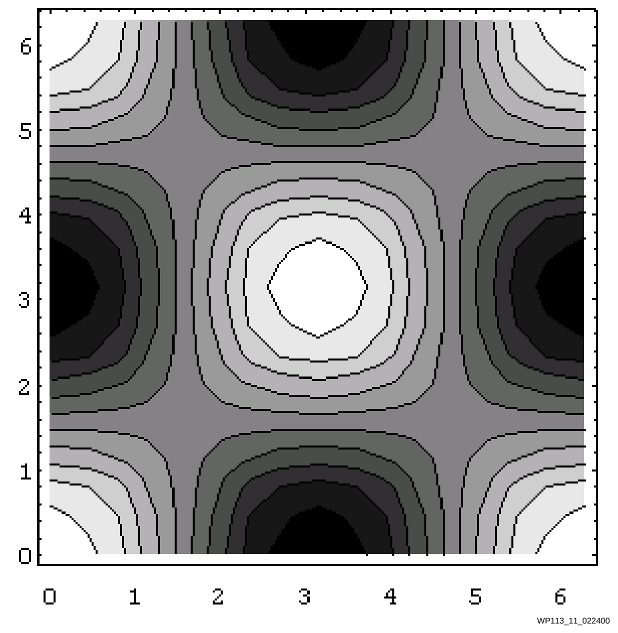

Figure 11: **Contour Plot of the Function f(x,y) = Cos(x) ∗ Cos(y)**

Approaches to 2-D FPGA Implementation

Two-dimensional DCT FPGA implementation can be done several ways. Three of these ways are:

- one dimensional row and one dimensional column DCT computations done in sequence;
- performing the one dimensional rows' DCTs in parallel and also, the one dimensional column DCT calculations also in parallel; and lastly,
- using the polynomial approach to represent the row and column values as coefficients of variables in polynomials.

The technique behind the separate one dimensional row and column computation relies on the fact that the two dimensional DCT can be reduced to a multiplication of two summations. That is, the two summations can be treated as two separate summations of the two cosines and then multiplied together. This calculation is of course obviously slow.

A way of speeding up the described sequential chain of events is by having a separate DCT processor for each row and each column, thereby having the ability to perform the row DCT transform in parallel and the same with the column DCT transforms. An unfortunate obvious result of this is that this takes more of the FPGA resources compared to the simpler approach previously but the speedup is still very useful.

The third method is much more efficient. Figure 12 shows the block diagram of the polynomial transform DCT.[3]

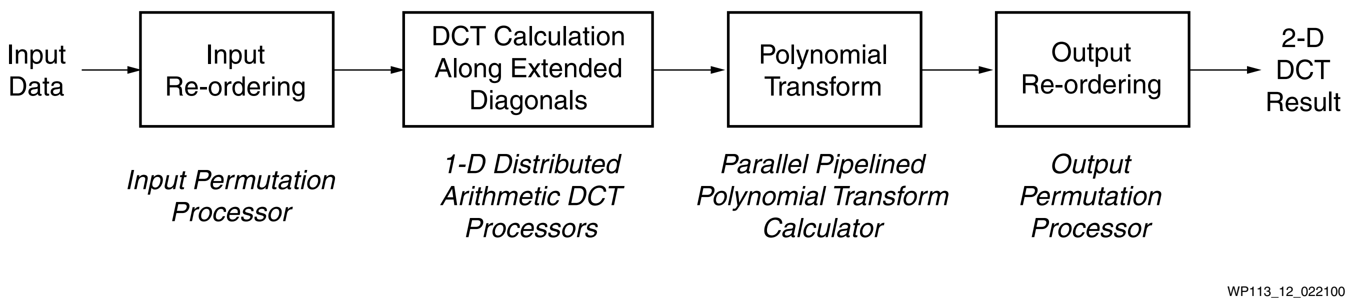
Input Data
Input Re-ordering
DCT Calculation Along Extended Diagonals
Polynomial Transform
Output Re-ordering
2-D DCT Result
Input Permutation Processor
1-D Distributed Arithmetic DCT Processors
Parallel Pipelined Polynomial Transform Calculator
Output Permutation Processor
WP113_12_022100

Figure 12: **Polynomial Transform DCT System Architecture**

The advantage of the polynomial transform DCT method is that multiplications are eliminated which in effect removes the computational noise inherent in rounding calculations. Another advantage is that this DCT architecture uses only 62% of the CLB resources of the Distributed Arithmetic approach for the same throughput.

Traditionally, high-end consumer applications like set-top boxes and digital cameras use 32-bit embedded controllers. In these devices, the DCT and IDCT transforms are implemented oftentimes in software. Figure 13 highlights the XIlinx advantage.

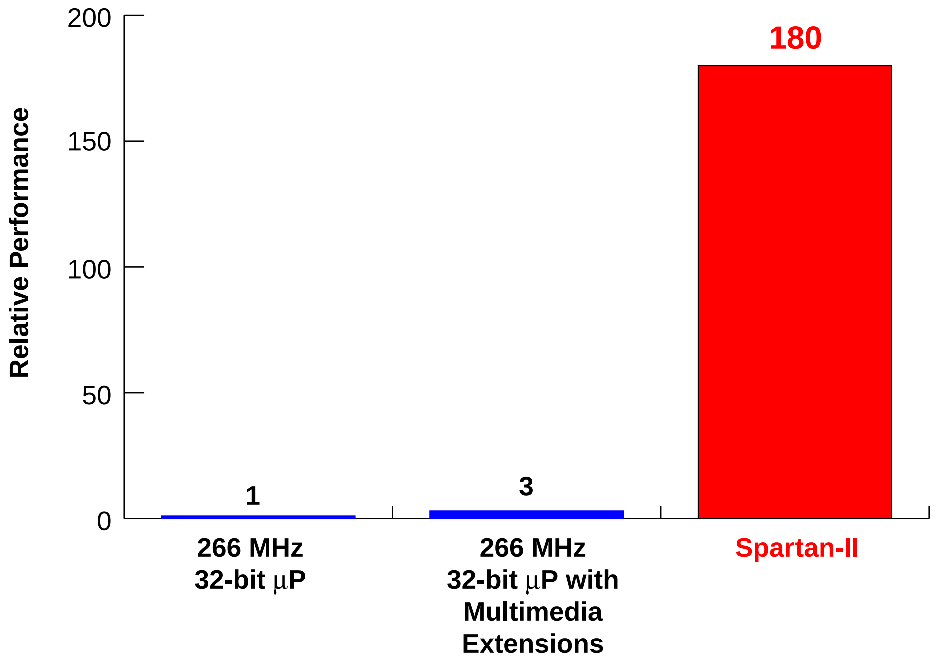

Figure 13: **Xilinx DCT/IDCT Solution Compared to a 32-bit µP Software Solution**

Summary

As exhibited, DCT and IDCT are indeed computationally intensive and are quite easily handled by Xentec's synthesizable solution. This solution takes advantage of silicon reuse by using a mode pin that switches the core to a DCT or an IDCT functionality and vice versa. Xilinx provides this high-performance solution as a part of its AllianceCORE offering.

Having the programmable ASSP solution, Xilinx provides not only the capabilities to compete in the presently evolving market but also at future evolving markets. In a very practical sense, the programmable ASSP roadmap provided by Xilinx, future proofs all products and applications that incorporate the Spartan-II family of FPGAs as well as other Xilinx programmable ASSP solutions.

Compared to traditional ASSP solutions for DCT and IDCT, the Xilinx solution provided by Xentec provides both DCT and IDCT in one synthesizable high-performance core that fits in 85% of a Spartan-II XC2S100 with enough BRAMs (block RAMs) and slices to implement several high-speed buffers. This DCT/IDCT core operates at 33.3 MHz and can easily flip between DCT and IDCT with the use of one mode pin. This definitely promotes silicon reuse as well as efficient use of board space.

Conclusion

The Spartan-II based solution offers higher throughput and flexibility in comparison to embedded software-based DCT/IDCT implementations.

Spartan-II FPGAs offer more than 100,000 system gates at under $10 and are the most cost-effective solution ever offered. They build on the capabilities of the very successful Virtex family and supports all the associated features including SelectI/O™ BlockRAM, Distributed RAM, DLLs, and support clock speeds up to and including 200 MHz. Spartan-II extends the Spartan family focus in replacing inflexible ASICs and traditional ASSPs.

References

1. *Digital Signal Processing A Practical Approach* by Ifeachor and Jervis
2. *C Language Algorithms For Digital Signal Processing* by Embree and Kimble
3. *Minimum Multiplicative Complexity Implementation of the 2-D DCT using Xilinx FPGAs* by Chris Dick
4. *Video Demystified* by Keith Jack
5. *Xentec DCT/IDCT Datasheet*
6. *Efficient Quadtree Coding of Images and Video* by Gary Sullivan and Rich Baker, ICASSP 91, pp 2661-2664.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
2/25/00	1.0	Initial Xilinx release.