An Inverse Discrete Cosine Transform (IDCT) Implementation in Virtex for MPEG Video Applications

XAPP208 (v1.1) December 29, 1999

Application Note: K. Chaudhary, H. Verma and S. Nag

Summary

This application note describes an implementation of IDCT in the Virtex™ family. DCT/IDCT are used in the MPEG video standard to reduce the bandwidth requirements. IDCT is one of the most computation-intensive parts of the MPEG decoding process. A fast, hardware based IDCT implementation is crucial to speed the MPEG decoding process. In this implementation, the inherent parallelism is exploited to achieve throughput as high as 3.28 Gbits/s, making it suitable for real time video applications. The implementation is synthesizable Verilog code at the RTL level.

Introduction

MPEG stands for Moving Picture Expert Group. The MPEG video compression standard is used in many current and emerging products. It is at the heart of digital television, set-top boxes, DSS, HDTV decoders, DVD players, video conferences and Internet video.

Video applications need compressed information. A typical HDTV standard for broadcasting in the US is 1,920 by 1,080 pixels at the rate of 30 frames/sec. Assuming eight bits for each of the three primary colors in a pixel, the required transmission rate is 1.5 Gbit/sec. Bandwidth limitations allow the transmission bandwidth for the video portion of the signal to be 18 Mbit/sec per channel. This translates to a required compression ratio of 83:1. Considering the end-user demand for very high quality video, the ratio is daunting. Compressed video information is useful for both bandwidth and storage constraints.

MPEG video is divided into a hierarchy of layers. From the top level, the first layer is the video sequence layer. This layer is defined as any self-contained bitstream, like a coded movie or advertisement. The next layer is a series of frames. Each frame comprises a set of macroblocks. Each macroblock comprises a set of blocks. A block is composed of pixels. We know from research in Human Visual Systems that the eye is most sensitive to changes in luminance and less to changes in chrominance. So, instead of the RGB color-space, the YC_BC_R color-space is used where Y is the luminance signal, C_B the blue-color difference signal, and C_R the red color difference signal. Reducing information in the C_B and C_R elements has less of an effect on picture quality than reducing information in the Y portion. This fact is used in MPEG video where different bit-level accuracy may be used for the Y and C_BC_R components.

In general, there is a high correlation between neighboring pixels in an image. Therefore, the Discrete Cosine Transform, an invertible transform, can be used to concentrate information into fewer decorrelated parameters. A block of size 8x8 pixels is transformed to produce 8x8 DCT coefficients. Out of the 64 coefficients, most of the higher order coefficients are small.

.

36	32	26	19	14	11	10	10		75	39	21	0	0	0	0	0
32	29	22	16	11	9	8	8	DCT	42	23	0	0	0	0	0	0
26	22	17	11	7	5	4	5	⇒	19	0	0	0	0	0	0	0
19	16	11	6	2	1	1	2		0	0	0	0	0	0	0	0
14	11	7	3	0	0	0	1	⇐	0	0	0	0	0	0	0	0
12	9	5	2	0	0	2	3	IDCT	0	0	0	0	0	0	0	0
11	9	5	2	1	2	4	6		0	0	0	0	0	0	0	0
11	9	6	3	3	4	6	8		0	0	0	0	0	0	0	0

Figure 1: The Effect of a Discrete Cosine Transform

The example in Figure 1 shows the original information for luminance (shown on the left) transformed to new coefficients (shown on the right) with just six out of 64 coefficients being non-zero. Also, all the zero coefficients are grouped towards the bottom-right. The top-left elements represent low-order coefficients and the bottom-right elements represent high-order coefficients.Typically, the bottom right coefficients are small, but not necessarily zero.

The DCT process is used in conjunction with a quantization matrix. The quantization matrix has larger numbers in the bottom right. Dividing the transformed coefficients with the corresponding entry in the quantization matrix favors zeroing out the bottom-right elements. The quantization matrix is dynamically updated to meet the objective of minimal picture degradation achievable for an allowed bit-rate.

The DCT and IDCT equations are shown below. The original value is f(x,y). F(μ,ν) is the DC-transformed value. F(0,0) is simply the summation of the f(x,y)'s and is thus the DC value. F(0,1) is a correlation of f(x,y) values with a cosine wave of one cycle in the y dimension.

$$C(\mu, \nu) = \frac{1}{4} C(\mu) C(\nu) \quad (1)$$

$$\psi(x, \mu) = \cos\left[\frac{(2x+1)\mu\pi x}{16}\right] \quad (2)$$

$$F(\mu, \nu) = C(\mu, \nu) \sum_{x=0}^{7} \sum_{y=0}^{7} f(x, y)\psi(x, \mu)\psi(y, \nu) \quad (3)$$

$$C(\mu) = \frac{1}{\sqrt{2}} \; for \; (\mu = 0)$$

$$C(\mu) = 1 \; for \; (\mu = 1, 2, \ldots 7)$$

$$f(x, y) = \sum_{\mu=0}^{7} \sum_{\nu=0}^{7} C(\mu, \nu) F(\mu, \nu)\psi(x, \mu)\psi(y, \nu) \quad (4)$$

Use-case Scenario

This Virtex based implementation has a huge speed advantage (3.28 Gbits/s) over software and DSP processor based implementations, thereby making good quality, real-time video a possibility on a PC. The use of this design is envisioned in the following scenario: A user has an FPGA-based PC-card used for a myriad of applications (MPEG-decoding, DES/MD5-based encryption/decryption, speech analyzer). Each application has a tremendous utility in this age of explosive internet growth. ASIC based implementations require an ASIC for each application and also a hardware replacement for any updates. The FPGA card based methodology provides cost effective flexibility for both upgrades and new applications. The application based FPGA reprogramming can be made transparent to the end user with a web-browser plug-in model.

Implementation Details

This section describes the details for IDCT implementation in Virtex FPGAs. 2D IDCT deals with the variation in both x and y dimensions. It is defined in terms of two 1D IDCTs, one for the y dimension and the other for the x dimension. F(u,v) denotes DCT coefficients used to reconstruct f(x,y); the original pixel value. The derivation for representing 2D IDCT in terms of two 1D IDCTs is shown in the following equations. By combining the terms containing v and defining G(u,y), f(x,y) is written using G(u,y). Notice the similarity of the two equations representing 1D IDCTs.

Using the notation:

$$G(\mu, y) = \frac{1}{2}\sum_{\nu=0}^{7} C(\nu)F(\mu,\nu)\psi(y,\nu) \qquad (5)$$

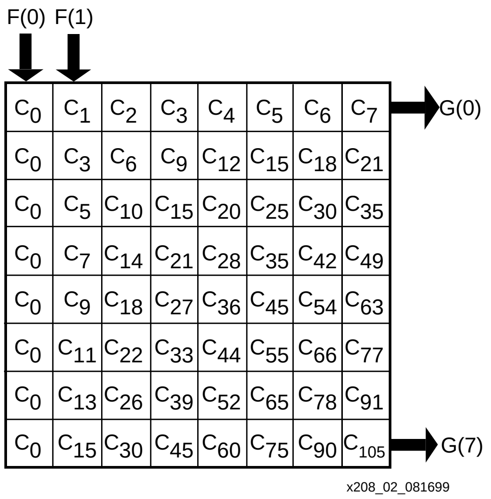
Rewrite Equation 4 as:

$$f(x, y) = \frac{1}{2}\sum_{\mu=0}^{7} C(\mu)G(\mu,y)\psi(x,\mu) \qquad (6)$$

Equation four shows the 2D IDCT performed using the two identical 1D IDCT modules in equations five and six. Next, a look at the implementation details of 1D IDCTs in Virtex devices. The 1D IDCT equations (equations 5 and 6) can be written in the simplified form shown in equation 7.

$$G(y) = \sum_{\nu=0}^{7} F(\nu)\cos\left[\frac{(2y+1)\nu\pi}{16}\right] \qquad (7)$$

This equation comprises a set of multiply-by-constant and add terms. It is ideally suited for distributed arithmetic [2], where partial solutions are stored in DALUTs (Distributed Arithmetic Look-Up Tables). The two parts of this equation are the F(v) part defined as either the DCT coefficients F(u,v) or the 1D IDCT output G(u, y); in either case, it is just a stored value. The other more interesting part is the cosine term. This equation is pictorially shown in Figure 2. Inputs for each column F(0) etc., get multiplied by the corresponding cosine terms C_0 etc., to produce the outputs G(0) etc., for each row. C_k is a concise definition for the cosine term cos(kπ/16).


F(0)
F(1)
G(0)
G(7)
x208_02_081699

C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7
C_0	C_3	C_6	C_9	C_{12}	C_{15}	C_{18}	C_{21}
C_0	C_5	C_{10}	C_{15}	C_{20}	C_{25}	C_{30}	C_{35}
C_0	C_7	C_{14}	C_{21}	C_{28}	C_{35}	C_{42}	C_{49}
C_0	C_9	C_{18}	C_{27}	C_{36}	C_{45}	C_{54}	C_{63}
C_0	C_{11}	C_{22}	C_{33}	C_{44}	C_{55}	C_{66}	C_{77}
C_0	C_{13}	C_{26}	C_{39}	C_{52}	C_{65}	C_{78}	C_{91}
C_0	C_{15}	C_{30}	C_{45}	C_{60}	C_{75}	C_{90}	C_{105}

Figure 2: 1D IDCT Equation Coefficients

C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7
C_0	C_3	C_6	$-C_7$	$-C_4$	$-C_1$	$-C_2$	$-C_5$
C_0	C_5	$-C_6$	$-C_1$	$-C_4$	C_7	C_2	C_3
C_0	C_7	$-C_2$	$-C_5$	C_4	C_3	$-C_6$	$-C_1$
C_0	$-C_7$	$-C_2$	C_5	C_4	$-C_3$	$-C_6$	C_1
C_0	$-C_5$	$-C_6$	C_1	$-C_4$	$-C_7$	C_2	$-C_3$
C_0	$-C_3$	C_6	C_7	$-C_4$	C_1	$-C_2$	C_5
C_0	$-C_1$	C_2	$-C_3$	C_4	$-C_5$	C_6	$-C_7$

x208_03_081699

Figure 3: Simplified Coefficients

Any C_k can be equated to C_P where:

P = [k mod 16] - 16 [(k mod 16) mod 8].

Examples; $C_{10} = -C_6$ and $C_{105} = -C_7$.

Using this, the simplified set of coefficients are shown in Figure 3

There is a useful symmetry in these new coefficients. For example, the odd entries of the 3rd row and 6th row are the same in magnitude but the opposite in sign; the even entries are identical. Using this symmetry, the 64 coefficients form eight groups of four entries each. Figure 4 illustrates a block diagram implementation in a Virtex device using DALUTs. The DALUTs are addressed by F(0) through F(7). The even and odd inputs are separated and their partial products (stored in the DALUTs) are added to form the eight outputs G(0) through G(7). The computation can be done either in a bit-serial fashion or a bit-parallel fashion. G(0) through G(7) are computed for the 12 bits of the F inputs in bit-serial fashion to reduce the area requirements.

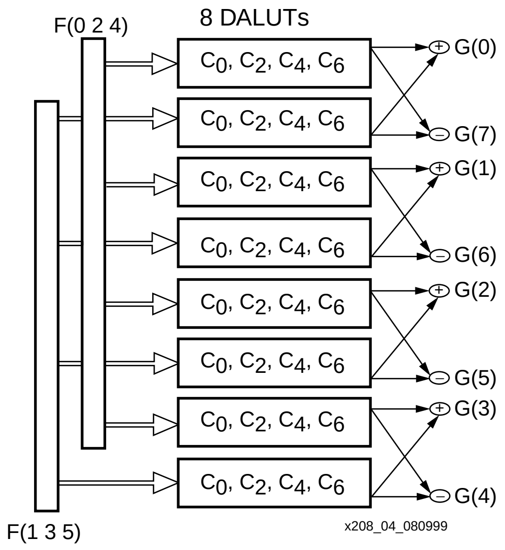


Figure 4: Block Diagram of Reference Design Implementation

In this implementation there are 16 entities of the block shown in Figure 4, eight for row computations and eight for column computations. The synthesizable Verilog code reference design, xapp208.zip, is on Xilinx's web site at:

http://www.xilinx.com/techdocs/htm_index/app_xapp.htm

The block in Figure 4 corresponds to module one_d_idct. To process a 8x8 data block, 13 clock cycles are required for row computations (12 clock cycles for bit serial computation and one clock cycle for handshaking), and another 13 for column computations. This results in a latency of 26 clock cycles. Row computation and column computation stages are pipelined. A new data block can be fed every 13 clock cycles.

Top Level Module Description

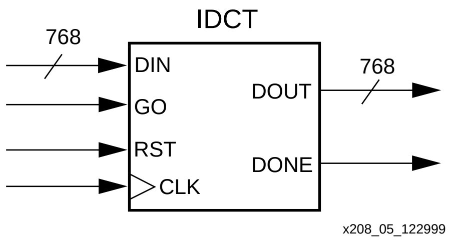


Figure 5: IDCT Top Level Module

The input, DIN, is a bit vector of size 768 obtained by concatenating a 8x8 matrix of 12-bits wide DCT coefficients in two's complement form. LSB of the input data from row i and column j is DIN(12 x [i + j]) and the MSB is DIN(12 x [i + j +11]). Similarly, the output DOUT is a bit vector obtained by concatenating the output from the IDCT. A High signal at RST pin resets all the flip-flops in the design. A pulse at the GO pin starts the IDCT computation. This pulse is given when the input data is valid. The DONE pin is asserted High when the computation of a block is over and the data on DOUT pin is valid. The timing diagram is shown in Figure 6.

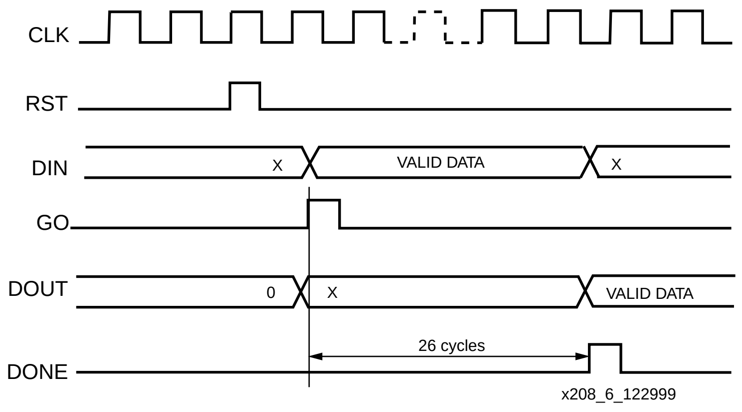


Figure 6: Timing Diagram

Implementation

Table 1: Results

	Exemplar	Synplicity
Part	XCV600 BG560-5	XCV600 BG432-5
Utilization	88% (6140 slices)	95% (6617 slices)
Performance	55.6 MHz	53.2 MHz
Latency	467.9 ns	489 ns
Throughput (data blocks/sec)	4.27 M	4.09 M
Throughput (bits/sec)	3.28 G	3.14 G

The results in Table 1 include two extra modules to reduce external pin requirements. The Verilog code for these modules are not included in the reference design since the system level requirements typically determine their design. A serial-to-parallel module at the input converts 16 sets of 48 bits to a 768-bit bus. A parallel-to-serial module at the output converts a 768-bit IDCT output into 16 sets of 48-bits each.

Conclusion

In this application note, an IDCT implementation based on the Virtex family is described. The inherent parallelism in the transformation algorithm has been exploited to achieve a throughput of 3.28 Gbits/sec. This high speed can be used to provide good quality, real-time video for internet applications.

The Verilog code can be found as a reference design file entitled xapp208.zip at:

http://www.xilinx.com/techdocs/htm_index/app_xapp.htm

References

1. J. Wiseman, *An Introduction to MPEG Video Compression*, DSP World, 1998, pp. 45-66.
2. Les Mintzer, *The role of Distributed Arithmetic in FPGAs*, available at:

   http://www.xilinx.com/appnotes/theory1.pdf

Revision History

Date	Version #	Revision
08.31.99	1.0	Initial release.
12.29.99	1.1	Reformatted from initial release.