Design and Implementation of Transform and Quantization Blocks of H.264 Video Encoder using VHDL and MATLAB

Satheesh Kumar S[1], Suchendranath Popuri[2], Rajeev Pankaj Nelapati[3]
School of Electronics Engineering, VIT University, Vellore, India.

ABSTRACT

This paper proposes an FPGA implementation of Integer transform and quantization blocks of H.264 video encoder. The algorithm and architecture of the components of the integer transformation, quantization was developed, designed and coded in VHDL. The above specified blocks of H.264 were coded in Matlab in order to verify the results of VHDL implementation. The processor is implemented on a Xilinx Virtex-2P XCV30 FPGA. And the implemented blocks occupied around 4322 slices out of 13696, 7391 LUTs out of 27392, 89 IOBs out of 416 and achieved speed of 103MHz.

Keywords: Integer transforms, quantization, video encoder, VHDL.

1 INTRODUCTION

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.

The invention of H.264 (MPEG-4 part 10) video encoding technology has been met with great enthusiasm in the video industry [1]. H.264 has video quality similar to that of MPEG-2, but is more economical with its use of bandwidth. Being less expensive to distribute, H.264 is a natural choice for broadcasters who are trying to find cost effective ways of distributing High Definition Television (HDTV) channels and reducing the cost of carrying conventional Standard Definition channels [2]. In fact, the use of bandwidth has been reduced to the point that it has captured the interest of telephone and data services providers, whose bandwidth limited link to the subscriber had previously not allowed for delivery of bandwidth thirsty television services. H.264 has the potential to revolutionize the industry as it eases the bandwidth burden of service delivery and opens the service provider market to new players. revolutionize the industry as it eases the bandwidth burden of service delivery and opens the service provider market to new players.

The main target of H.264 standard is reducing the clock rates and power consumption to make the design suitable for battery powered video applications, and also double the coding efficiency (which means having the bit rate necessary for a given level of fidelity) in comparison to any other existing video coding standards, while at the same time sharing common features with existing standards. In this work, we have realized some of the above features such as 4x4 integer transform and quantization. The implementation conforms to the base line main as well as extended profiles since only intra frames are used [9].

This paper is organized as follows: we will detail about the Integer transform and Quantization blocks in Section 2, section 3 deals with the architecture of the proposed schemes .Simulation results are presented in Section 4 and Conclusions are presented in section 5.

2 INTEGER TRANSFORM AND QUANTIZATION

The two dimensional DCT (2-D DCT) is a mathematical tool that is used to transform the information from the space domain to the frequency domain. In image and video compression, many standards like JPEG, MPEG and H.264/AVC use the 2-D DCT to transform the input data to the frequency domain. The information represented in the frequency domain could be handled to discard the frequencies that are less important to the human visual system perception [4]. This operation reduces the amount of information used to represent the image or video, allowing higher compression rates with little impact in the image quality. There are many image and video compression standards that apply the 2-D DCT over 8x8 input blocks. However the DCT used in the H.264/AVC is applied over 4x4 input blocks, reducing the computational complexity of this calculation [5]. The H.264/AVC is the first video standard that uses an integer transform.

Regarding the transform used, two issues of interest are: the block size, and the transform itself [10]. In general, larger the block size used for transform, better it would be for exploiting global correlations, on the other hand, the smaller the block size used for transform, better it would be for exploiting local adaptivity in content.

The disadvantage of the DCT, which is used in the earlier standards, is that the entries of the transforms are irrational numbers. Thus when we compute the direct and
inverse transform in cascade we may not get exactly the same data back . Keeping in view the drift discussed above a 4x4 integer transform, which is the orthogonal approximation to DCT, allowing for bit-extract implementation for all encoder and decoders was proposed. H.264 is unique that it employees this purely integer spatial transform as opposed to the usual floating point 8x8 DCT specified with rounding error tolerance as used in earlier standards [12]. The small shape helps to reduce the blocking and ringing artifacts, while the precise integer specification eliminates any mismatch between the encoder and decoder in the inverse transform. The multiplications by 1/2 in the inverse transform can be implemented by the sign preserving1-bit right shifts thus reducing the complexity. The forward (C_f) and inverse (C_i) transforms are given below.

$$C_f = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \qquad C_i = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1/2 & -1/2 & -1 \\ 1 & -1 & -1 & 1 \\ 1/2 & -1 & 1 & -1/2 \end{bmatrix}$$

The forward transform is applied as

`

$$W = C_f X C_f^T . \tag{1}$$

And the inverse transform is applied as

$$W_i = C_i X_a C_i^T . \tag{2}$$

Where X_q are the quantized and scaled samples. The quantization stage removes less important information, making it possible to compress the remaining data. The key to process is that, whilst the original information co-efficient may take on a large number of possible values and hence the reconstructed co-efficient are restricted to the discrete set of values.

2.1 QUANTIZATION

Quantization is basically a process for reducing the precision of the DCT coefficients, precision reduction is extremely important. Since lower precision almost always implies a lower bit rate in the compressed data stream [3]. H.264 assumes a scalar quantizer [11]. The mechanisms of the forward and inverse quantizers are complicated by the requirements to (a) avoid division and/or floating point arithmetic and (b) incorporate the post and pre-scaling matrices E_f and E_i described above. The basic forward quantizer operation is:

$$Z_{ij} = round(W_{ij} / Q_{step}). \tag{3}$$

where w_{ij} is a co-efficient of the transform described above, Q_{step} is a quantizer step size and Z_{ij} is a quantized coefficient. The rounding operation here need not round to the nearest integer, for example, biasing the `round' operation towards smaller integers can give perceptual quality improvements. A total of 52 values of Q_{step} are supported by the standard, indexed by a quantization parameter (QP) as shown in Table 1. Q_{step} doubles in size for every increment of 6 in QP. The wide range of quantizer step sizes makes it possible for an encoder to control the trade off accurately and flexibly between bit rate and quality. The values of QP can be different for luma and chroma. Both parameters are in the range 0 to 51 and the default is that the chroma parameter [2]. The post scaling factor a2, ab/2 or b2/4 is incorporated into the forward quantizer. First, the input block X is transformed to give a block of un-scaled coefficients $w=C_fXC_f^T$, then each coefficient W_{ij} is quantized and scaled in a single operation:

$$Z_{ij} = round(W_{ij}(PF / Q_{step})). \tag{4}$$

PF is a^2, ab/2 or b^2 /4 depending on the position (i,j)

Position	PF
(0,0),(2,0),(0,2) or (2,2)	a^2
(1,1), (1,3), (3,1) or (3,3)	$b^2/4$
Other	ab/2

The factor PF/Q_{step} is implemented in the reference model software as a multiplication by a factor MF and a right shift, avoiding any division operations:

$$Z_{ij} = round(W_{ij}(MF / 2^{qbits})). \tag{5}$$

Where

$$MF / 2^{qbits} = PF / Q_{step}. \tag{6}$$

And

$$qbits = 15 + floor(QP / 6). \tag{7}$$

In integer arithmetic equation can be implemented as follows

$$Z_{ij} = W_{ij}.MF + f >> qbits. \tag{8}$$

$$sign(Z_{ij}) = sign(W_{ij}). \tag{9}$$

where >> indicates a binary shift right. In the reference model software, f is 2^{qbits}=3 for Intra blocks or 2^{qbits}=6 for Inter blocks.

QP	0	1	2	3	4	5	6
Q_{step}	.625	.6875	.8125	.875	1	1.125	1.25
QP	7	8	9	10	11	12	…
Q_{step}	1.375	1.625	1.75	2	2.25	2.5	…
QP	…	18	…	24	…	30	…
Q_{step}	…5	5	…	10	…	20	…
QP	36	…	42	…	48	…	51
Q_{step}	40	…	80	…	160	…	224

Table 1. Quantization step size for H.264 video codec

2.2 RESCALING

The basic scaling (or `inverse quantiser') operation is:

$$W'_{ij} = Z_{ij}.Q_{step}. \qquad (10)$$

The pre-scaling factor for the inverse is incorporated in this operation, together with a constant scaling factor of 64 to avoid rounding errors:

$$W'_{ij} = Z_{ij}.Q_{step}.PF.64. \qquad (11)$$

3 DESIGN AND IMPLEMENTATION OF TRANSFORM AND QUANTIZATION BLOCKS

This section introduces the proposed VLSI prototype of the 4x4 forward transform adopted by the H.264 standard. The proposed architecture uses 4x4 parallel input blocks, a block diagram of the architecture is shown in figure 1.

This block consists of two cascaded sub-blocks, each of them is responsible for multiplying (Multiplier 1 and Multiplier 2) two 4x4 matrices and is composed of four identical butterfly adder blocks. Its operation is to perform a group of additions and shifts as shown in figure 2.

Figure 2(a) shows the first buttery adder block in the first sub-block, while Figure 2(b) shows the first buttery adder block in the next sub-block. The signals S0 - S9 are control signals used to control add and shift operations. The inputs to `MUL1' are the residual pixels in a column of the residual block (X). The four columns of X are every clock cycle; a column of X is applied at the inputs of MUL1. The four columns are given as inputs four times [12] . The partial products are generated in the order F00, F01, F02, F03, ... F32, F33. As soon as four new partial products are generated, the store block stores these outputs for the next four clock cycles.

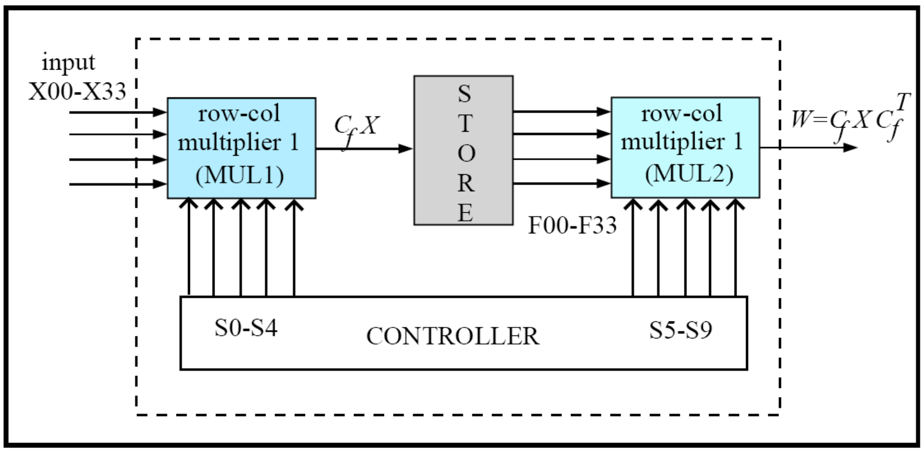

Fig.1.Architecture of core forward integer transform

The four partial product outputs (one row of matrix F) from the store block are inputs to the second row column multiplier (MUL2). These partial product outputs are held constant for four clock cycles during which they are multiplied with each of the columns of C_f^T, by changing the control signals to MUL2 appropriately to produce the transformed coefficients serially.

After the DCT calculation, the transformed output then sent to quantization block, quantization is basically a process for reducing the precision of the DCT coefficients, precision reduction is extremely important. H.264 assumes a scalar quantizer. Quantization in H.264 is arithmetically expressed as a two-stage operation. The first stage is multiplying each coefficient in the 4x4 block by a fixed coefficient specific value. This stage allows the coefficients to be scaled unequally according to importance or information. The second stage is dividing by an adjustable quantization parameter (QP) value. This stage provides a single “knob" for adjusting the quality and resultant bit-rate of the encoding. The two operations can be combined into a single multiplication and single shift operation. The QP is expressed as an integer from 0 to 51. This integer is converted to a quantization step size (Q_{step}) nonlinearly. Each 6 steps increases the step size by a factor of 2, and between each pair of power-of-two step sizes N and 2N there are 5 steps: 1.125N, 1.25N, 1.375N, 1.625N, 1.75N.

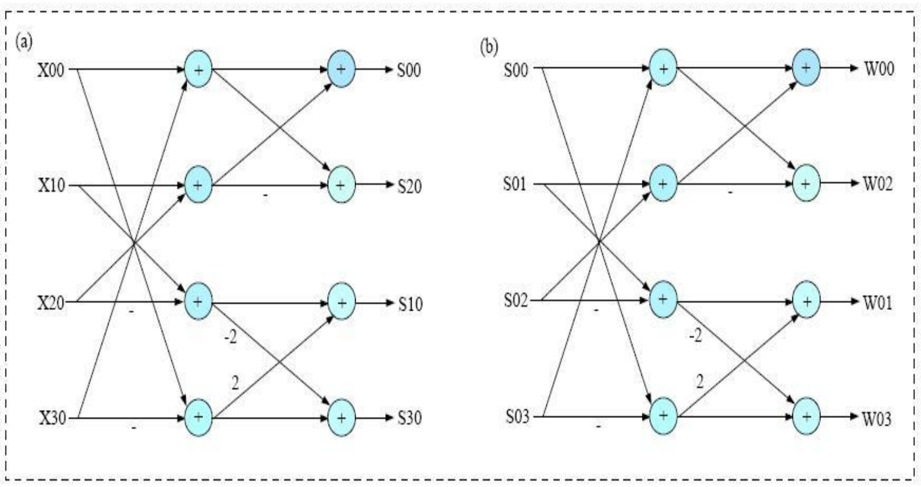

Fig. 1. First buttery adder block in (a) first sub-block (b) second sub block

4 SIMULATION RESULTS

The various blocks outlined in section 3 are described in MATLAB and VHDL and theimulated in Modelsim simulator. A few of simulated results are depicted in Fig 3 to Fig 9.

4.1 MATLAB RESULTS

The transformation and quantization coefficients of original “Lena" image have been computed using MATLAB. The input image of “Lena" and output images of (DCT and iDCT)”Lena" for different Quantization Parameters (QP=5, QP=20 and QP=40) using MATLAB are shown in Figure 3 to 5

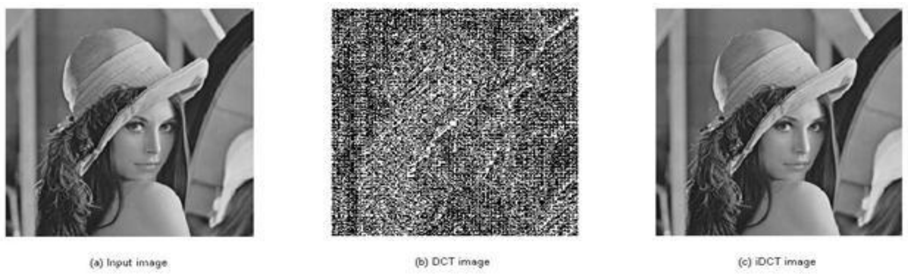

Fig. 2. (a)Input image (b)DCT image (c)iDCT image

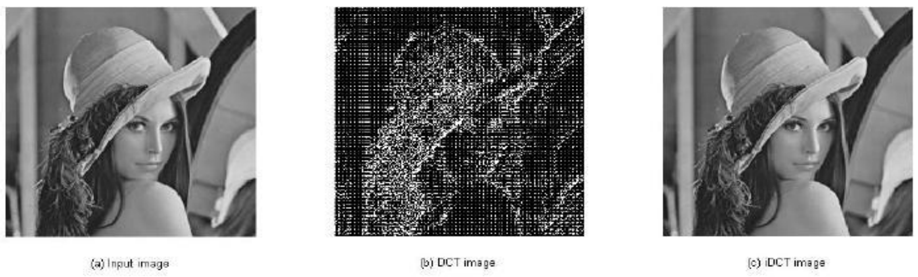

Fig. 3. (a)Input image (b)DCT image (c)iDCT image

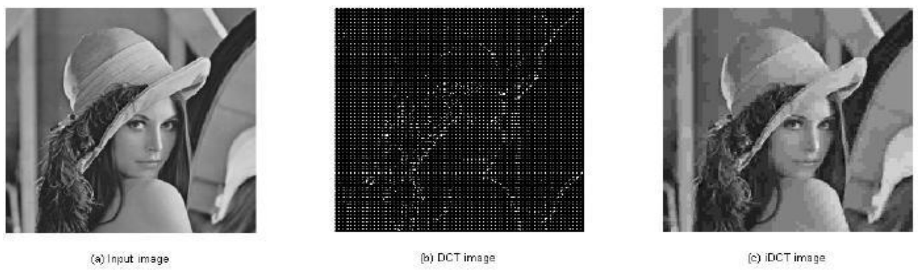

Fig. 4. (a)Input image (b)DCT image (c)iDCT image

It has been found from the experiments that, for QP = 5 the output coefficient given by MATLAB are almost equal to the input coefficient. As the value of QP increases compression increases so the value of PSNR decreases accordingly the picture quality reduces.

$$MSE = \frac{1}{MN}\sum_{X=1}^{M}\sum_{Y=1}^{N}[I(x,y) - I'(x,y)]' \qquad (12)$$

$$PSNR = 20\log_{10}(255/\sqrt{MSE}) \qquad (13)$$

where I(x,y) is the original image, I'(x,y) is the decompressed image and M, N are the dimensions of the images. For different quantization parameters the PSNR values shown in the table 2 below:

QP	PSNR
5	56.23
10	51.64
15	47.66
20	43.45
30	35.83
40	29.44
50	22.64

Table 2. PSNR output for different QP values

4.2 VHDL RESULTS

VHDL simulation results for various QP values were simulated. From the above simulations for QP=10 the simulation results of Transform, Quantization and Inverse transformation are depicted below:

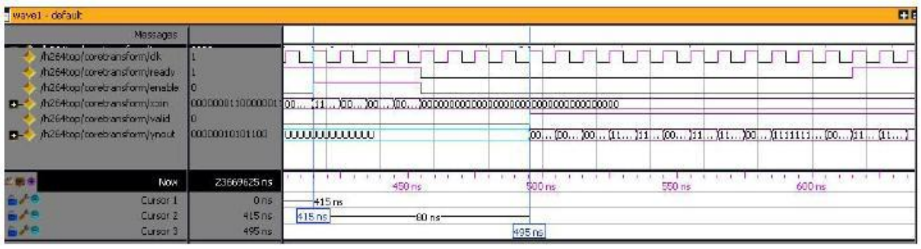
wave1 - default
Messages
/h264top/coretransform/clk
/h264top/coretransform/ready
/h264top/coretransform/enable
/h264top/coretransform/xxin
/h264top/coretransform/valid
/h264top/coretransform/ynout
Now
Cursor 1
Cursor 2
Cursor 3
415 ns
80 ns
495 ns

Fig. 5. Simulation results of H.264 Core Transformation for QP=10

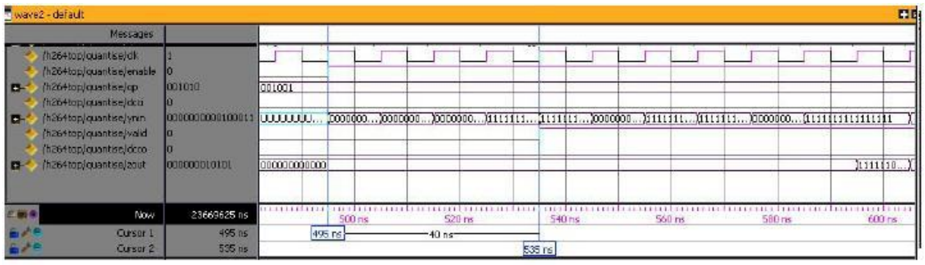

Fig. 6. Simulation results of H.264 Quantization for QP=10

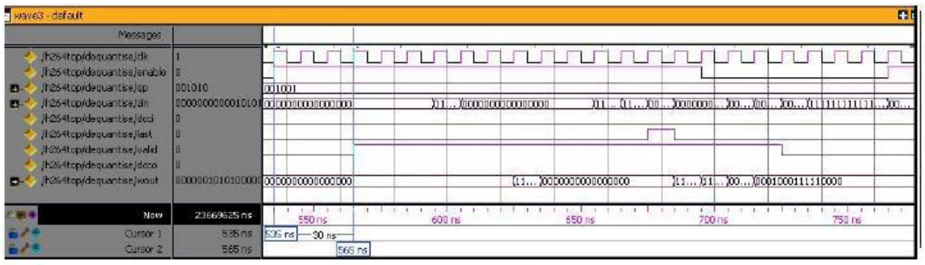

Fig. 7. Simulation results of H.264 Inverse Quantization for QP=10

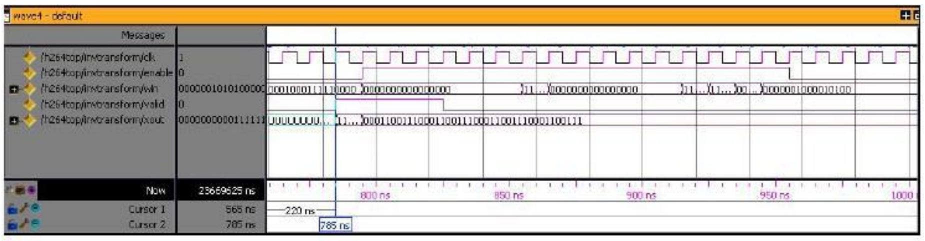
wave4 - default
Messages
/h264top/invtransform/clk 1
/h264top/invtransform/enable 0
/h264top/invtransform/win 000000101010000
/h264top/invtransform/valid 0
/h264top/invtransform/xout 000000000011111
Now 23669625 ns
Cursor 1 565 ns
Cursor 2 785 ns
220 ns
785 ns
800 ns
850 ns
900 ns
950 ns
1000

Fig. 8. Simulation results of H.264 Inverse Transformation for QP=10

After successful simulation of working of different blocks of H.264 encoder (transform, quantization, rescale and inverse transform) we then proceeded for synthesis of code on Xilinx Virtex-2P XCV30 FPGA. And also we have compared our synthesis results with the performance of existing implementations. Comparison shows that our design achieved higher frequency.

Summary of the synthesis results and timing reports are shown in Tables 3 and 4.

S .No		This work	Existing implementation [12]
1	No. of flip flops	65	257
2	No. of Slices	167	992
3	Max. freq	185MHz	103MHz

Table 3. Synthesis result comparison of core transform with existing implementation

S .No		This work	Existing implementation
1	No. of flip flops	355	608
2	No. of Slices	246	424
3	No. of LUTs	321	656
4	Max. freq	259MHz	189MHz

Table 4. Synthesis result comparison of quantization with existing implementation

5 CONCLUSIONS AND FUTURE WORK

An FPGA implementation of the core processors of H.264 video encoder, the major emphasis of the implementation was on the transformation and quantization, as the transformation and quantization requires 60 to 70 percent of the total time. The results were compared by considering the images which are of different sizes. The 4x4 integer transform used is significantly simpler and faster than the 8x8 DCT used in MPEG 2. A significant improvement over MPEG-2 is the reduction of blocking artifacts, especially for high compression. The processor is implemented on a Xilinx Virtex-2P XCV30 chip and the implemented blocks occupied around 4322 slices out of 13696, 7391 LUTs out of 27392, 89 IOBs out of 416 and achieved speed of 103MHz.

The present work can be extended to: FPGA Implementation of remaining blocks of H.264 Video Encoder i.e. Re-ordering and Entropy Coding and Reconstruction of frame using predicted image and integrating with the other blocks of the Encoder i.e. Motion Estimation and Motion Compensation.

REFERENCES

1 T. Wiegand, G. J. Sullivan, and Ajay Luthra, "Overview of the h.264/avc video coding standard," *IEEE Trans on Circuits and Systems for Video Tech*., vol. 13, no. 7, July 2007.

2 I. E. G. Richardson, Video Codec Design. *John Wiley and Sons Ltd*., 2002.

3 A. Ortega, and K. Ramchandran, "Rate-distortion methods for image and video compression: An overview," *IEEE Signal Processing magazine*, November 1998.

4 L. Agostini, R. Porto, M. Porto, T. Silva, L. Rosa and J. Guntzel, "Forword and inverse 2-d dct architectures targetting hdtv for h.264/avc video compression standard," *International Broadcasting Convention,* no. 37, pp. 11-16, 2007.

5 H. S. Malvar, A. Hallapuro, M. Karaczewicz, and L. Kerofsky, "Low-complexity transform and quantization in h.264/avc," *IEEE Trans. Circuits Syst. Video Technol*., vol. 13, no. 7, July 2003.

6 Roman C. Kordasiewicz, and S. Shirani, "Asic and fpga implementations of h.264 dct and quantization blocks," *McMaster University Department of Electrical and Computer Engineering Hamilton*, pp. 7803-9134, 2005.

7 M. E. Al-Mulla, C. N. Canagarajah, and D. R. Bull, "Video coding for mobile communications," *Academic Press*, 2002.

8 W. Huang, B. Y. Hsieh, T. C. Chen, and L. G. Chen, "Analysis, fast algorithm, and vlsi architecture design for h.264/avc intra frame coder," *IEEE Trans. Circuits System VideoTech*., vol. 15, no. 3, pp. 378-401, March 2005.

9 M. Horowitz, A. Joch, F. Kossentini, and A. Hallapuro, "H.264/avc baseline profile decoder complexity analysis," *IEEE Trans. Circuits Syst. Video Technol.,vol*. 13, no. 7, July 2003.

10 Minyi Fu, G. A. Jullien, V.S. Dimitrov, and M. Ahmadi, "A low-power dct ip core based on 2d algebraic integer encoding," *Proceedings of the 2004 International Symposium on Circuits and Systems*, vol. 2, pp. 757-765, may 2004.

11 I. Amer, W. Badawy, and G. Jullien, "Hardware prototyping for the h.264 4x4 transformation," *in Acoustics, Speech, and Signal Processing, ICASSP*, vol. 5, pp. 77-80, May 2004

12 L. Wang, Z. Zhang, and G. Teng, "Hardware implementation of transform and quantization for avs encoder," *IEEE Trans. Advanced Display and System Application, Ministry of Education*, pp. 4244-1724, July 2008.