Journal of Basic and Applied Engineering Research
Print ISSN: 2350-0077; Online ISSN: 2350-0255; Volume 2, Number 5; January-March, 2015, pp. 386-389
© Krishi Sanskriti Publications
http://www.krishisanskriti.org/jbaer.html

A Review on an Efficient Implementation of H.264 Video Encoder DCT Transform and Quantization

Hemika[1] and Poornima Sharma[2]

[1,2]M.Tech (VLSI Design) Banasthali University (Rajasthan)
E-mail: [1]hemikayadav90@gmail.com, [2]poornima.sharma829@gmail.com

Abstract—*H.264 is a digital video codec eminent for high data compression while maintaining high quality. Codec is usually used for videos uploaded to the web. It is the Part of the MPEG-4 codec. One of the very nice things about H.264 is that we can use it at very low and very high bitrates. In this paper, we present hardware efficient architecture for 8x8 transform and quantization for H.264 encoder. For transformation, 2D-DCT (discrete cosine transformation) is used. Quantization which is used to convert transform coefficient into integer value which is implemented in Verilog. To implement the architecture in Verilog HDL we used shifters instead of multiplier to reduce complexity of computation.*

Index terms*: DCT, H.264, and Quantization.*

1. INTRODUCTION

In today's trend, H.264 is a high definition digital video. It will send highly compressed low resolution video across the web and then encode high definition movie at super high bitrates for delivery to a High Definition television. This is very common codec for camcorders and digital video cameras. Its container is AVCHD (advanced video coding high definition). H.264/MPEG-4 part10 AVC is block oriented motion-compensation-based video compression standard developed by the ITU-T Video Coding Experts Group (VCEG) together with the ISO/IEC JTC1 Moving Picture Group (MPEG). The project

Corporation effort is known as the Joint Video Team (JVT).

Transform coding techniques have turn into the essential model in image and video coding standards, in which Discrete Cosine Transform (DCT).In Video encoder performs video data compression by combination of main modules such as Motion estimation and compensation, Transformation, Quantization and Entropy encoding. Among these modules, transformation is the module of removing the spatial redundancy that exists in the spatial domain of video sequence. Discrete Cosine Transformation (DCT) is the transformation method in existing image and video coding standards. This paper presents efficient implementation of DCT Transform.

In H.264, video is captured as a series of frames. Each frame is compressed by partitioning it as one or more slices, where each slices consists of sequence of macro blocks. These macro blocks are transformed and quantized. The transformation module converts the frame data from time domain to frequency domain, which intends to decor relate the energy i.e. amount of information present in the frame. Since the transformation module is reversible in nature, this process does not change the information content present in the source input signal during encoding and decoding process.

In fact, the initial specification adopted an integer approximation of 4x4[1]. But, the 4x4 block is not enough for SD resolutions and above. That is, when larger then 4x4 transforms are used, significant compression performance gains have been reported are Standard Definition (SD) and High Definition (HD) resolutions [2]. Thus, an 8x8 transform and quantization is represented here. Previous works have been successes in hardware implementation of transform and quantization. In [3], Proposed a design with a high through-put and low latency architecture using CSD multiplier for shared quantization inverse –quantization. However, the disadvantage is that their hardware implementation area could not be significantly reduced.

In this paper, efficient hardware 2D-DCT and quantization architecture which have minimum complexity for 8x8 with reduced parallelism. This is prepared as follows. Section 2 presents review of 2D-DCT Transform. Section 3 describes quantization used in H.264 encoder. Section 4 describes the proposed architecture and performance analysis; conclude in section 5, future work in section 6, acknowledgement in section 7 and references in section 8.

2. 2D-DCT TRANSFORM

The H.264 integer transform is a fundamental form of Discrete Cosine Transform: all operations can be performed by using shift, add and multiply arithmetic, it is possible to ensure zero mismatches between compressor and decompresses. The main part of transform can be done using additions and shifts. A scaling multiplication is integrated to the quantizer which decreases the total number of multiplications. In [9]-[11] proposed 8x8 integer approximation of DCT. It consists of add and shifts without any multiplication. The 4x4 DCT of an input array X is given by:

Y=AXAT i.e.

$$\begin{bmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{bmatrix} \begin{bmatrix} & & \\ & X & \\ & & \end{bmatrix} \begin{bmatrix} a & b & a & c \\ a & c & -a & -b \\ a & -c & -a & b \\ a & -b & a & -c \end{bmatrix}$$

Where
a = 1/2, b = 1/2cos π/8 or 2/5 and
c = 1 / 2 cos(3π/ 8)

Hardware architecture of 8x8 integers transform for H.264 which promises very low resources utilization. In the architecture, each pixel is processed one by one on a simplified pipeline without multiplication. Thus, undesired modules, which are used for block-based or row-based parallel processing, can be reduced.

The 2-D forward 8x8 transform is computed in a separable way as a 1-D horizontal (row) transform followed by a 1-D vertical (column) transform, where the corresponding 1-D transforms are given by Equation (1) and Matrix shown below,

$$W = CXC^T \qquad (1)$$

Matrix C 8x8 is specified as

$$\begin{bmatrix} 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 \\ 12 & 10 & 6 & 3 & 8 & 8 & -10 & -12 \\ 8 & 4 & -4 & -8 & -8 & -4 & 4 & 8 \\ 10 & -3 & -12 & -6 & 6 & 12 & 3 & -10 \\ 8 & -8 & -8 & 8 & 8 & -8 & -8 & 8 \\ 6 & -12 & 3 & 10 & -10 & -3 & 12 & -6 \\ 4 & -8 & 8 & -4 & -4 & 8 & -8 & 4 \\ 3 & -6 & 10 & -12 & 12 & -10 & 6 & -3 \end{bmatrix} .1/8$$

The architecture of 1-D integer transform is shown in Fig. 1. Each input column vector of 8 pixels is input to the 1-D DCT block for 8 cycles, so transformed outputs w0-w7 are also held for 8 cycles. However, just one out of w0-w7 is output through MUX. That is, w0, w1…... w7 go out of the 1-D DCT in sequential order for 8 cycles. After all, 64 cycles are required to process all pixel elements in one 8x8 block.

Each of these 1-D transform can be computed via fast butterfly operations as follows:

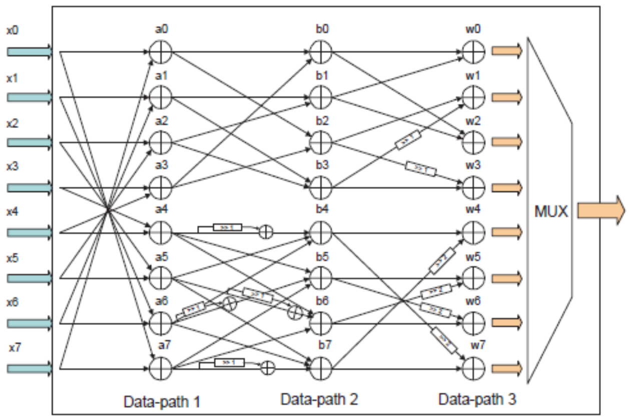

Fig. 1: 1D Row (column) transform block

Data-path 1:
a[0] = x[0]+x[7]
a[1] = x[1]+x[6]
a[2] = x[2]+x[5]
a[3] = x[3]+ x[4]
a[4] = x[0] –x[7]
a[5] = x[1]-x[6]
a[6]= x[2]-x[5]
a[7]= x[3]-x[4]

Data-path 2:
b[0]= a[0]+a[3]
b[1]= a[1]+a[2]
b[2]= a[0]-a[3]
b[3]= a[1]-a[2]
b[4]= a[5]+a[6]+((a[4]>>1)+a[4])
b[5]= a[4]-a[7]-((a[6]>>1)+a[6])
b[6]= a[4]+a[7]-((a[5]>>1)+a[5])
b[7]= a[5]-a[6]+((a[7]>>1+a[7])
Data-path 3:
w[0]= b[0]+b[1]
w[2]= b[2]+(b[3]>>1)
w[4]= b[0]-b[1]
w[6]= (b[2]>>1)-b[3]
w[1]= b[4]+(b[7]>>2)
w[3]= b[5]+(b[6]>>2)
w[5]= b[6]-(b[5]>>2)
w[7]= -b[7]+(b[4]>>2)

3. QUANTIZATION

H.264 standard supports both 4x4 and 8x8 quantization. Quantization reduces the precision of the transform coefficients according quantization parameter. Original coefficient values is divided by a QP and rounded to the nearest integer. Setting QP to high means that more coefficients are set to zero, resulting in high compression but degrade quality.

Quantization and scaling is performed according to the following equation:
$Z_{ij=}$ round (W_{ij} x PF/Qstep)

Where PF is a2, ab/2 or b2/4 depending on the position (i,j). For position (0, 0), (2, 0), (0, 2), or (2, 2) PF is a2. For (1, 1), (1, 3), (3, 1) or (3, 3) PF is b2/4 and other position PF will be ab/2. The factor (PF/Qstep) is implemented in H.264 as multiplication factor (MF) depending on the position of the coefficient.
Zij= round (Wij x MF/2qbits) where
MF/2qbits=PF/Qstep and qbits =16+n
Therefore, equation can be written as:
Zij= (Wij*MF+ f * 216+n) >> (16+n)

Where n is QP/6. QP is the quantization parameter which specifies MFs, and f is the dead zone/offset parameter with an absolute value ranging between 0 and 1/2 and with the same sign as the coefficient that is being quantized. QP is the quantization parameter ranging from 0 to 51.

In case of H.264 as specified in, multiplication factor MF depends on m(=QP/6) and the position(i,j) of the element. Total of 52 values of Qstep are supported by the standard and these are indexed by QP. The values of Qstep corresponding to each QP are shown in table 1. Note that, Qstep doubles in size for every increment of 6 in QP and Qstep increases by 12.5% for each increment of 1 in QP.

Table 1: Quantization step sizes in H.264

QP	0	1	…..	4	….	11	12
Qstep	0.625	0.6875	….	1	…	…	25
QP	….	18	….	..	…	…	51
Qstep	….	5	….	..	…	…	224

This standard specifies the first six values of MF as shown in Table 2. For QP>5 the factor MF remains unchanged but the value of qbits will change. For example, qbits=16 for $0 \leq QP \leq 5$ (as qbits =16 +QP / 6), qbits =17 for $6 \leq QP \leq 11$ and so on.

4. PROPOSED WORK

The block diagram of the architecture for 8×8 integer transform and quantization is presented in Figure2, which contains 1-D DCT transform blocks, transpose logic, quantization block and control logic.In this system word length is 16-bit. This architecture clearly reflects two separate 1-D integer transforms and quantization. In the architecture, each pixel is processed one by one on a simplified pipeline without multiplication. The pixel by- pixel processing can remove unnecessary modules used for block-based or row-based processing in not only the integer transform block but the quantization block.

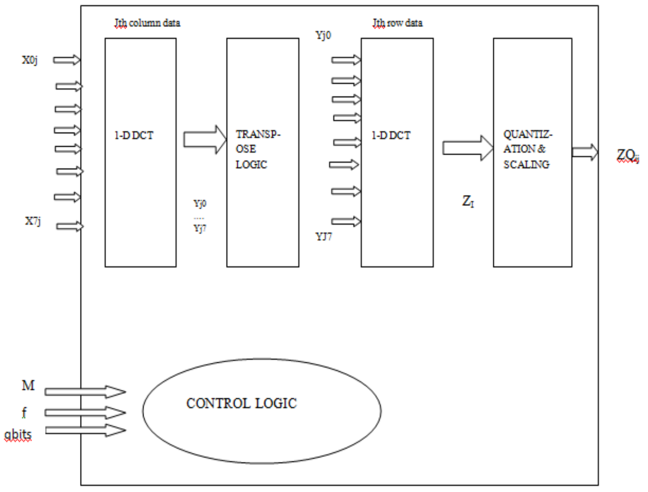

Fig. 2: Architecture of transform and quantization

5. CONCLUSION

Here we represented a review of an efficient implementation of a 2d-DCT transform and quantization for H.264. It supports most of video formats which satisfying real-time constraints. In this implementation, each pixel is processed by pipelined and this pipelined structure can eliminate unnecessary modules used for row-based processing. To implement the architecture in Verilog HDL, we used shifters instead of multiplier to reduce the computation time.

6. FUTURE WORK

We will synthesis this architecture in xilinx Virtex IV.After observing the synthesis results we will further try to reduce area occupied by our architecture, to improve speed and to reduce power consumption.

7. ACKNOWLELDGEMENT

To determine, analyze and to present something new is to venture on an untraded path towards an unexplored destination is an arduous adventure unless one gets a true torchbearer to show the right way. We would have never succeeded in completing our task without the support, encouragement and help provided to us by various people. The enlightening guidance, we found in our revered guide Mr. Dharmendra Kumar, Senior Design Engineer, Silicon Mentor, and Greater Noida, without whose patronization it was never possible to give final shape to this project. We wish to express our deep gratitude towards him for providing individual guidance.

REFRENCES

[1] T. Wiegand, G. J. Sullivan, G. Bjontegaard and A. Luthra , "Overview of the H.264/AVC Video Coding Standard,"*IEEE Trans. Circuits Syst. Video Technol.*, Vol. 13, no. 7, pp. 560-576, July 2003.

[2] S. Gordon, D. Marpe, and T.Wiegand, "ABT for Film Grain Reproduction in High Definition Sequences,"*Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG,* doc. JVT-H029, Geneva, Switzerland, May 2003.

[3] C.-P. Fan and Y.-L. Cheng, "FPGA implementations of low latency and high throughput 4x4 block texture coding processor for H.264/AVC," *Journal of the Chinese Institute of Engineers,* vol. 32, no. 1, pp. 33–44, 2009.

[4] H. Malvar, A. Hallapuro, M. Karczewicz, and L. Kerofsky, "Low-complexity transform and quantization in H.264/AVC," *IEEE Transactions on Circuits and Systems for Video Technology,* pp. 598–603, 2003.

[5] J.-K. Lee and K.-D. Chung, "DCT block conversion for H.264/AVC video transcoding," *in Euro-Par 2005 Parallel Processing, Lisbon, Portugal, September 2005,* pp. 919–927.

[6] Jeoong Sung Park and Tokunbo Ogunfumi, "A New Hardware Implementation Of the H.264 8x8 Transform And Quantization," *IEEE* 978-1-4244-2354-5/09/$25.00 ©2009.

[7] R. Korah, J. Raja Paul Perinbam, "FPGA implementation of integer transform & quantizer for H.264 Encoder," *Journal of VLSI signal processing systems - Springer*, 2008.