Journal of Theoretical and Applied Information Technology
30th April 2012. Vol. 38 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

EFFICIENT DEISGN OF LOW AREA BASED H.264 COMPRESSOR AND DECOMPRESSOR WITH H.264 INTEGER TRANSFORM

[1]KALIKI SRI HARSHA REDDY, [2]R.SARAVANAN

[1]M.Tech VLSI Design, SASTRA University, Thanjavur, Tamilnadu, India

[2]Assistant Professor, School of Computing, SASTRA University, Thanjavur, Tamilnadu, India

[1]Email: sriharshakaliki@gmail.com, [2]saravanan23378@gmail.com

ABSTRACT

The Video Compressor/Decompressor (CODEC) performs compression and decompression of video data by combining a Motion estimation and compensation module, Transformation module, and Entropy encoding module. Among these three modules, the spatial redundancy can be eliminated by the transformation module which exists in the spatial domain of video sequence. Discrete Cosine Transformation (DCT) is the transformation method in existing image and is a popular transformation technique for intra and inter frame video coding. This paper presents an architecture for transformation, inverse transformation, quantization and inverse quantization modules for one macro block (16 x 16 pixel region with one luma and two chroma components) with low area and minimum number of clock cycles using integer transform. H.264 Integer transform is a modified form of Discrete Cosine Transform. Using ASIC design tools a lossless design of 8 x 8 and 4x4 transforms were implemented with integer transform. The improved area and speed performance results are presented.

Keywords- *H.264 Compressor/Decompressor, Integer Transform, Discrete Cosine Transformation (DCT), Macro Block*

1. INTRODUCTION

The two most important benefits of video compression are First; with video compression it is possible to use digital video in transmission environments where uncompressed video is not applicable. The second benefit is, video compression enables more efficient use of transmission resources. Video compression can be achieved by eliminating the components that is not useful for reproduction of data. The two types of redundancies that present in video data are spatial and temporal. Spatial redundancy refers to the correlation between pixel values which are close to each other. Removal of spatial redundancy can be done by looking into the frame and is hence referred to as intra frame coding. Temporal redundancy, on the other hand is the redundancy present in neighbouring frames. Video sequence at high frame rate normally has high correlation among the consecutive frames. Thus removal of such temporal redundancy involves correlation among the frames and is called inter frame coding. Spatial redundancy is removed by using transform coding techniques. Temporal redundancy can be eliminated by using motion estimation and compensation techniques. In H.264/Advanced Video Coding (AVC**)**, video is interpreted as a series of frames. Each frame can be compressed by splitting it into one or more slices, where each slice contains a series of macro blocks. These macro blocks are transformed quantized and encoded. The transformation module alters the frame data from time domain to frequency domain, which intends to decorrelate the energy (i.e., amount of information present in the frame) present in the spatial domain. It also alters the energy components of the frame into small numbers of transform coefficients, which are more efficient for encoding rather than their original 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. By information theory, transformed coefficients are reversible in nature. There have been many explorations for designing the H.264

Compressor/Decompressor. Compressor and two Decompressors were designed on the same chip to support HDTV functionality. Along with the H.264 codec a RISC processor is also integrated on the same chip. Compressor/Decompressor architecture is proposed for the mobile device. There have been lots of thrust among the researcher to minimize the area of H.264 c this paper uses the integer transform to develop a low area based design and to improve the speed of H.264 Compressor and Decompressor. This will be extremely useful for the mobile devices. Section 2 and Section 3 of the paper gives functionality of H.264 compressor and decompressor, section 4 discuss about the Integer transform, section 5 and 6 covers the Decompressor and Compressor flow respectively ,section 7 and 8 discusses about the multipliers and hardware architecture respectively and the section 9 presents the Theoretical calculations and section 10 presents the results.

2. FUNCTION OF H.264 COMPRESSOR

H.264 Compressor contains two data paths, one is a 'forward' path (left to right) and other is a 'reconstruction' path (right to left).The

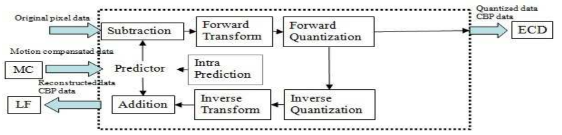


Figure.1. *Function of H.264 Compressor*

dataflow path in the Decompressor is indicated from right to left is to show the similarities between Compressor and Decompressor. The predicted frame in the forward path of Compressor that comes from motion compensation is subtracted from the current block to produce a residual block which is transformed and quantized to give a set of quantized transform coefficients which are reordered and entropy encoded. In the reconstruction path the decompressor decodes (reconstructs) it to provide a reference for further predictions. The coefficients are scaled and inverse transformed (T^{-1}) to produce a residue. The prediction block PRED is added to create a reconstructed block. The effects of distortion can be reduced with low pass filter (LPF) and with a series of blocks reconstructed picture is created.

3. FUNCTION OF H.264 DECOMPRESSOR

The decompressor receives a compressed bit stream and entropy decodes the data elements to produce a set of quantized coefficients X. These coefficients were inverse quantized and inverse transformed. The prediction block is generated by the Decompressor using the header information present in the bit stream, which is similar to the prediction PRED formed in the Compressor. The reconstruction block is produced by adding PRED which is filtered to generate each decoded block.

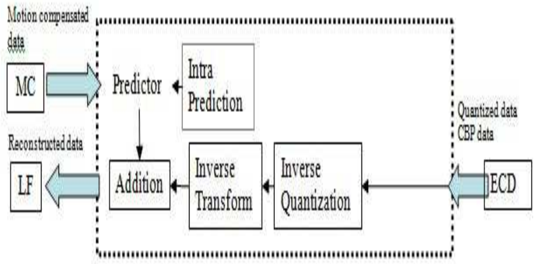


Figure.2. *Function of H.264 Decompressor*

4. INTEGER TRANSFORM

The H.264 integer transform is a fundamental form of Discrete Cosine Transform:
1. In an integer transform all operations can be performed by using shift, add and multiply arithmetic.
2. It is possible to ensure zero mismatches between Compressor and Decompressor.
3. The main part of transform can be done using only additions and shifts.
4. A scaling multiplication (part of the transform) is integrated to the quantizer which decreases the total number of multiplications.
The forward Transform can be represented as:

$$Y = (A \text{ X } A^T) \text{ x } E \qquad (1)$$

Where X is a pixel matrix,Y the result matrix and A is the transform matrix. E is the scaling matrix.
The inverse transform can be represented as:

$$X=A^T(Y\text{x}E)A \qquad (2)$$

$$A_{iJ} = C_i\cos(2j+1)i\pi/2N \qquad (3)$$

$$\text{Where:}C_i=1/N^{1/2}(i=0) \qquad (4)$$

$$C_{i=}2/N^{1/2}(i>0) \qquad (5)$$

4.1 Derivation procedure

The Transform matrix for 4x4 pixel region

$$\mathbf{A} = \begin{bmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & c \end{bmatrix} \qquad \text{where } \begin{array}{l} a = \frac{1}{2} \\ b = \sqrt{\frac{1}{2}}\cos\left(\frac{\pi}{8}\right) \\ c = \sqrt{\frac{1}{2}}\cos\left(\frac{3\pi}{8}\right) \end{array}$$

is given as

The output representation using transform is given as

$$\mathbf{Y} = \mathbf{A}\mathbf{X}\mathbf{A}^{\mathrm{T}} = \begin{bmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{bmatrix} \begin{bmatrix} & & & \\ & \mathbf{X} & & \\ & & & \end{bmatrix} \begin{bmatrix} a & b & a & c \\ a & c & -a & -b \\ a & -c & -a & b \\ a & -b & a & -c \end{bmatrix}$$

where:

$$a = \frac{1}{2}, \qquad b = \sqrt{\frac{1}{2}}\cos\left(\frac{\pi}{8}\right), \qquad c = \sqrt{\frac{1}{2}}\cos\left(\frac{3\pi}{8}\right)$$

The final transform is reduced to the following equivalent form where CXC^{T} is a 'main' 2D transform and the symbol$\otimes$ indicates that each element of final transform is multiplied by the scaling factor (i.e corresponding elements multiplication)**.**The values of the constants a and b are as before and d is c/b (approximately 0.414) which is approximated to 0.5.

$$Y = (CXC^{\mathrm{T}}) \otimes E = \left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & d & -d & -1 \\ 1 & -1 & -1 & 1 \\ d & -1 & 1 & -d \end{bmatrix} \begin{bmatrix} & & & \\ & \mathrm{X} & & \\ & & & \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & d \\ 1 & d & -1 & -1 \\ 1 & -d & -1 & 1 \\ 1 & -1 & 1 & -d \end{bmatrix} \right) \otimes \begin{bmatrix} a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \\ a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \end{bmatrix}$$

The value of b is modified for so that b=√2/5
After substituting the values in the transform the second and fourth rows are scaled by a factor of two and the scaling matrix is also scaled to avoid the unnecessary multiplications so that there is no loss of accuracy .
The final forward transform is shown below

$$Y = C_f X C_f^{\mathrm{T}} \otimes E_f = \left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \begin{bmatrix} & & & \\ & \mathrm{X} & & \\ & & & \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 & 1 \\ 1 & 1 & -1 & -2 \\ 1 & -1 & -1 & 2 \\ 1 & -2 & 1 & -1 \end{bmatrix} \right) \otimes \begin{bmatrix} a^2 & \frac{ab}{2} & a^2 & \frac{ab}{2} \\ \frac{ab}{2} & \frac{b^2}{4} & \frac{ab}{2} & \frac{b^2}{4} \\ a^2 & \frac{ab}{2} & a^2 & \frac{ab}{2} \\ \frac{ab}{2} & \frac{b^2}{4} & \frac{ab}{2} & \frac{b^2}{4} \end{bmatrix}$$

By using forward and inverse Transform matrices the total function of Compressor and Decompressor is implemented. There are two 1D Transforms in the design, one is horizontal transform (operating row wise) and other one is vertical transform (operating column wise). In the horizontal transform of decompressor there are three stages in 8x8 region and two stages in 4x4 region. In the vertical transform of decompressor there are three stages in 8x8 region and two stages in 4x4 region. In the horizontal transform of compressor there are three stages in 8x8 regions and one stage in 4x4 region. In the vertical transform of compressor there are three stages in 8x8 region and one stage in 4x4 region. The entire process is pipelined. The multiplication and addition process will is independent of each other and the result will be stored in registers. In this process each element is considered as 16 bit binary number hence for an 8x8 to complete there are 64 16-bit registers. With the help of same architecture four 4x4 regions can be completed i.e. one chroma component can be completed with one 8x8 architecture. Hence the two chroma components can be completed with two 8x8 architectures, i.e.; four 4x4 regions at a time. The overall architecture is implemented by using bit serial technique.

5. DECOMPRESSOR FLOW

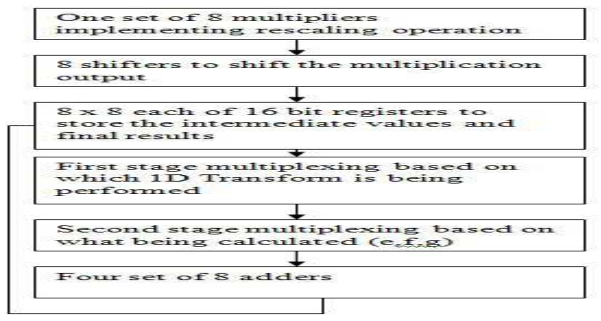

One set of 8 multipliers implementing rescaling operation
8 shifters to shift the multiplication output
8 x 8 each of 16 bit registers to store the intermediate values and final results
First stage multiplexing based on which 1D Transform is being performed
Second stage multiplexing based on what being calculated (e,f,g)
Four set of 8 adders

Figure.3.*Hardware* implementation of decompressor

One set of multipliers is used to complete the inverse quantization. Shifters were used to implement the shifting operation in Decompressor and registers were used to store the intermediate values. The adders were used to perform addition operation i.e. to implement the intermediate stages.

6. COMPRESSOR FLOW

The same hardware is used to implement the Compressor just by interchanging the connections. One set of multipliers is used to implement the quantization. Registers were used to store the intermediate values. The adders were used to perform addition operation i.e. to implement the intermediate stages.

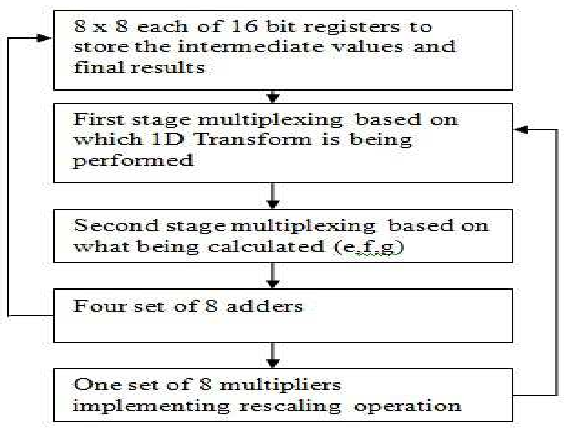

8 x 8 each of 16 bit registers to store the intermediate values and final results
First stage multiplexing based on which 1D Transform is being performed
Second stage multiplexing based on what being calculated (e,f,g)
Four set of 8 adders
One set of 8 multipliers implementing rescaling operation

Figure.4.*Hardware implementation of compressor*

7. MULTIPLIERS

There are eight multipliers used in the design to implement quantization and inverse quantization .Each multiplier will operate on one particular register at a time, hence eight multipliers will operate on one row or one column depending decompressor or compressor respectively. The output of multiplier is stored in corresponding registers. The entire process is pipelined hence the multiplication and addition can be independent of each other.

8. HARDWARE

The hardware used to implement both compressor and decompressor is same. With the help of control signals either compressor or Decompressor can be performed.. The only change in the compressor or Decompressor is pipelined diagram i.e. either adder will operate first or multipliers will operate first. For a decompressor multiplication will start first followed by addition parallel. In the compressor addition will start first followed by multiplication in parallel because of pipelining.

9. CYCLES FOR 1080P30 RESOLUTION:

9.1Theoritical

In 1080P30 resolution there are 30 frames, each frame consists of 8100 macro blocks, and hence total number of clock cycles for a frequency of 1GHZ is shown below.

For 1 GHZ:

1 macro block = $1 \times 10^9 / (30 \times 8100)$ (6)
= 4115 cycles.

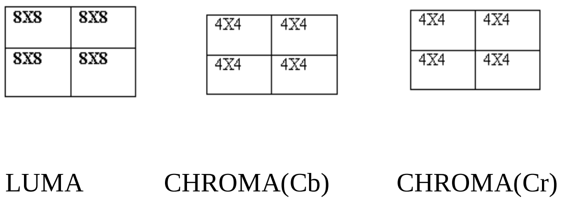
Table.1. *Cycles count*

FREQUENCY	CYCLES
1 GHZ	4115
1.1 GHZ	4527
1.2 GHZ	4938

10. EXPERIMENTAL REULTS

The experimental results were obtained using ASIC Design Tools Verilog Compiler Simulator (VCS) and Design Compiler (DC).The area report is obtained using Design compiler and the cycles count information is obtained using VCS.

MACRO BLOCK



In a macro block there are two components one is luma which is 16x16 region and two chroma components which are two 8x8 regions.

Hence the total number of clock cycles obtained for encode and decode process is

Total cycles for Compressor

$(4 \times 284) + (182 \times 2) + (4 \times 305) + (254 \times 2) =$ 3228cycles (7)

Total cycles for decompressor

$(4 \times 305)+ (254 \times 2) =$ 1728cycles (8)

10.1 AREA REPORT

By mapping the decompressor and compressor to the same design the total area of the design is found to be

Frequency	Total Area(gates)
1 GHZ	21862.1
1.1 GHZ	24123.3
1.2 GHZ	27206.6

Table.2.*Area report*

11. CONCLUSION

In this paper an energy efficient design for a low area based compressor and decompressor is presented. The architecture is based on the integer transform method which can be performed using shift and additions. The entire architecture is pipelined; hence the total number of clock cycles was reduced. The architecture provides a break-through by reducing 20 percent of the clock cycles and thereby increases its speed. From the experimental results it is observed that, even though the frequency of design is changing the area is with in limit.

REFERENCES

[1] Ian Richardson E.G.,H.264 and MPEG-4 Video compression – Video coding for next generation multimedia Wiley edition, The Robert Gordon University, Aberdeen UK(2003).

[2] ITUT-T Recommendation H.264 Advanced video coding for generic audio-visual Series H: Audio-visual and multimedia systems, pp.205-209.

[3] Mizosoe, H.; Yoshida, D. ; Nakamura, T. ; A SingleChipH.264/AVCHDTVEncoder/Decoder/Transcoder System LSI. International Conference on Consumer Electronics, 2007. ICCE 2007. Digest of Technical Papers. pp 1-2,

[4] Qiang Peng; Jin Jing; “H.264 codec system-on-chip design and verification “5th International Conference on ASIC, 2003. Proceedings. Publication Year: 2003, Page(s): 922 - 925 Vol.2

[5] Issa, S.; Khalifa, O.O.; Performance analysis of Dirac video codec with H.264/AVC. 2010 International Conference on Computer and Communication Engineering (ICCCE), Publication Year: 2010

[6] Zhenyu Wei; Kai Lam Tang; Ngan, K.N.; “Implementation of H.264 on Mobile Device”IEEE Transactions on Consumer Electronics, Volume: 53 , Issue: 3 Publication Year: 2007 , Page(s): 1109 – 1116