International Journal of Current Engineering and Technology
E-ISSN 2277 – 4106, P-ISSN 2347 – 5161

Available at http://inpressco.com/category/ijcet

Research Article

H.264 Video Encoding Standard

Anu Rachel Roy†* and Rincy Merin Varkey†

†Department of Electronics & Communication Engg,, St. Joseph's College of Engineering & Technology, Palai, Kerala, India

Accepted 28 Nov 2015, Available online 08 Dec 2015, **Vol.5, No.6 (Dec 2015)**

Abstract

H.264/AVC is newest video coding standard of the ITU-T Video Coding Experts Group and the ISO/IEC Moving Picture Experts Group. The main goals of the H.264/AVC standardization effort have been enhanced compression performance and provision of a network-friendly video representation addressing conversational (video telephony) and non conversational (storage, broadcast, or streaming) applications. H.264/AVC has achieved a significant improvement in rate-distortion efficiency relative to existing standards. For huge systems like video processing, FPGA prototyping plays an important role before taping out. In this paper, a verification system for H.264/AVC encoders with FPGA prototyping is proposed and implemented.

Keywords: *H.264, Motion Estimation, Run Length Encoding, DCT.*

1. Introduction

DCTs are important to numerous applications in science and engineering, from lossy compression of audio (e.g. MP3) and images (e.g. JPEG) (where small high-frequency components can be discarded), to spectral methods for the numerical solution of partial differential equations. The use of cosine rather than sine functions is critical for compression, since it turns out (as described below) that fewer cosine functions are needed to approximate a typical signal, whereas for differential equations the cosines express a particular choice of boundary conditions. In particular, a DCT is a Fourier-related transform similar to the discrete Fourier transform (DFT), but using only real numbers. DCTs are equivalent to DFTs of roughly twice the length, operating on real data with even symmetry (since the Fourier transform of a real and even function is real and even), where in some variants the input and/or output data are shifted by half a sample. There are eight standard DCT variants, of which four are common. The most common variant of discrete cosine transform is the type-II DCT, which is often called simply the DCT, its inverse, the type-III DCT, is correspondingly often called simply the inverse DCT or the IDCT. Two related transforms are the discrete sine transform (DST), which is equivalent to a DFT of real and odd functions, and the modified discrete cosine transform (MDCT), which is based on a DCT of overlapping data.

2. System Description

Compression is a reversible conversion (encoding) of data that contains fewer bits. This allows a more efficient storage and transmission of the data. The inverse process is called decompression (decoding). Software and hardware that can encode and decode are called decoders. Both combined form a codec and should not be confused with the terms data container or compression algorithms.

Lossless compression allows a 100files, where a loss of information is a major damage. These compression algorithms often use statistical information to reduce redundancies. Huffman-Coding and Run Length Encoding are two popular examples allowing high compression ratios depending on the data. Using lossy compression does not allow an exact recovery of the original data. Nevertheless it can be used for data, which is not very sensitive to losses and which contains a lot of redundancies, such as images, video or sound. Lossy compression allows higher compression ratios than lossless compression. The block diagram for the H.264 encoder using only intra prediction is shown in Figure 1.

Our baseline architecture will only support intra prediction, since inter prediction is considerably more complex. The three main building blocks are the DCT, Quant, and Intra-Prediction blocks, which will each be discussed below. The purpose of the inverse blocks is to perform the same steps the decoder will perform, and therefore base encoding decisions on how accurate the decoding process will be. For the most part, the inverse operation is conceptually and structurally similar to the forward operation, so further discussion on the inverse will be minimal. This standard is used for the video compression. Videos consists of many frames. Selection block selects the inter frames and intra frames. Reference frame block selects the inter

*Corresponding author: **Anu Rachel Roy**

frames and the present frame block selects both inter and intra frames. Their difference are calculated by using the motion estimation block and the difference is calculated by using necessary methods. The output is in the time domain and inverse DCT is used to convert it into frequency domain. Then it is quantized to reduce the size of the value. It is then arranged by using zig-zag. Run length encoding and Huffman encoding are also size reduction techniques and the bit stream is generated.

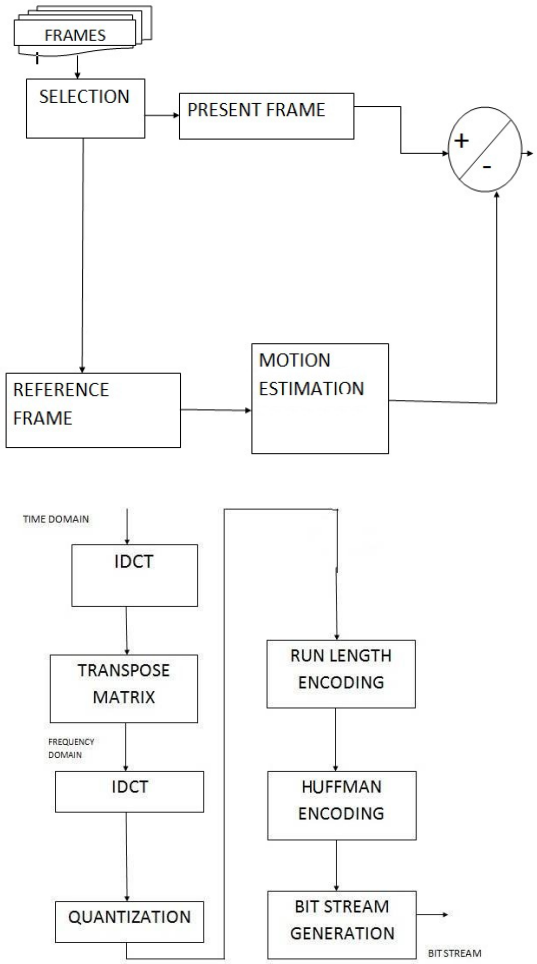

Fig. 1 H.264 Encoder

A. Selection or Intra Prediction

Intra-prediction utilizes spatial correlation in each frame to reduce the amount of transmission data necessary to represent the picture. H.264 performs intraprediction on two different sized blocks: 16x16 (the entire macroblock) and 4x4. 16x16 prediction is generally chosen for areas of the picture that are smooth. 4x4 prediction, on the other hand, is useful for predicting more detailed sections of the frame. The general idea is to predict a block, whether it be a 4x4 or 16x16 block, based on surrounding pixels using a mode that results in a prediction that most closely resembles the actual pixels in that block. There are nine 4x4 prediction modes, and four 16x16 modes. The four 16x16 modes are similar to modes 0, 1, 2, and a combination of modes 3 and 8 of the 4x4 modes. The intra-prediction block is a computationally intensive block. Making a prediction decision for a single 4x4 block requires: making a prediction for each pixel for each mode, computing the cost of using each prediction method, which includes calculating the sum of the differences (or sum of the hadamard transformed differences) between every prediction and actual pixel value, and 3) choosing the smallest cost as the correct prediction mode for that input block. 16x16 prediction is similar

B. Motion Estimation

Motion estimation is the process of determining motion vectors that describe the transformation from one 2D image to another; usually from adjacent frames in a video sequence. It is an ill-posed problem as the motion is in three dimensions but the images are a projection of the 3D scene onto a 2D plane. The motion vectors may relate to the whole image (global motion estimation) or specific parts, such as rectangular blocks, arbitrary shaped patches or even per pixel. The motion vectors may be represented by a translational model or many other models that can approximate the motion of a real video camera, such as rotation and translation in all three dimensions and zoom.

The motion estimation block in a video codec computes the displacement between the current frame and a stored past frame that is used as the reference. Usually the immediate past frame is considered to be the reference. More recent video coding standards, such as the H.264 offer flexibility in selecting the references frames and their combinations can be chosen. We consider a pixel belonging to the current frame, in association with its neighbourhood as the candidates and then determine its best matching position in the references frame. The difference in position between the candidates and its match in the reference frame is defined as the displacement vector or more commonly, the motion vector. It is called a vector since it has both horizontal and vertical components of displacement. We shall offer a more formal treatment to motion estimation in the next sections.

C. DCT

The DCT block takes in a 4x4 prediction residual and reduces the amount of redundancy by applying a transformation. The inverse DCT block, naturally, gets the necessary information back. The transformation used is a 4x4 integer transform that has all of the essential properties of the complex 8x8 DCT used by previous standards. The matrices used for the transformation and inverse transformation. Since the inverse transform is also integer, this transformation has the added benefit of having no encoder/decoder mismatch. The final output, Y , of the DCT block, given input X, is Y = HXHT.

A discrete cosine transform (DCT) expresses a finite sequence of data points in terms of a sum of cosine functions oscillating at different frequencies.

DCTs are important to numerous applications in science and engineering, from lossy compression of audio (e.g. MP3) and images (e.g. JPEG) (where small high-frequency components can be discarded), to spectral methods for the numerical solution of partial differential equations. The use of cosine rather than sine functions is critical for compression, since it turns out (as described below) that fewer cosine functions are needed to approximate a typical signal, whereas for differential equations the cosines express a particular choice of boundary conditions. In particular, a DCT is a Fourier-related transform similar to the discrete Fourier transform (DFT), but using only real numbers. DCTs are equivalent to DFTs of roughly twice the length, operating on real data with even symmetry (since the Fourier transform of a real and even function is real and even), where in some variants the input and/or output data are shifted by half a sample. There are eight standard DCT variants, of which four are common. The most common variant of discrete cosine transform is the type-II DCT, which is often called simply the DCT, its inverse, the type-III DCT, is correspondingly often called simply the inverse DCT or the IDCT. Two related transforms are the discrete sine transform (DST), which is equivalent to a DFT of real and odd functions, and the modified discrete cosine transform (MDCT), which is based on a DCT of overlapping data.

D. Quantization

This block first scales each transformed coefficient by a predefined value. It then quantizes each value for transmission. There are 52 different quantization levels possible, specified by the quantization parameter (QP). An increase in QP by six doubles the quantization step size, which doubles the compression. Thus, this block directly determines the compression versus quality tradeoff. A magnitude of the sampled image is expressed as a digital value in image processing. The transition between continuous values of the image function (brightness) and its digital equivalent is called quantitation. The number of quantitation levels should be high enough for human perception of fine shading details in the image. The occurrence of false contours is the main problem in image which have been quantized with insufficient brightness levels. This effect arises when the number of brightness levels is lower than that which humans can easily distinguish. This number is dependent on many factors, for example, the average local brightness – but displays which avoids this effect will normally provide a range of at least 100 intensity levels. Most digital image processing devices use quantization into k equal intervals.

E. Run Length encoding

Run-length encoding (RLE) is a very simple form of data compression in which runs of data (that is, sequences in which the same data value occurs in many consecutive data elements) are stored as a single data value and count, rather than as the original run. This is most useful on data that contains many such runs. Consider, for example, simple graphic images such as icons, line drawings, and animations. It is not useful with files that don't have many runs as it could greatly increase the file size. RLE may also be used to refer to an early graphics file format supported by CompuServe for compressing black and white images, but was widely supplanted by their later Graphics Interchange Format. RLE also refers to a little-used image format in Windows 3.x, with the extension rle, which is a Run Length Encoded Bitmap, used to compress the Windows 3.x startup screen. Typical applications of this encoding are when the source information comprises long substrings of the same character or binary digit.

F. Huffman Encoding

A Huffman code is an optimal prefix code found using the algorithm developed by David A. Huffman while he was a Ph.D. student at MIT, and published in the 1952 paper A Method for the Construction of Minimum-Redundancy Codes. The process of finding and/or using such a code is called Huffman coding and is a common technique in entropy encoding, including in lossless data compression. The algorithm's output can be viewed as a variable-length code table for encoding a source symbol (such as a character in a file). Huffman's algorithm derives this table based on the estimated probability or frequency of occurrence (weight) for each possible value of the source symbol. As in other entropy encoding methods, more common symbols are generally represented using fewer bits than less common symbols. Huffman's method can be efficiently implemented, finding a code in linear time to the number of input weights if these weights are sorted. However, although optimal among methods encoding symbols separately, Huffman coding is not always optimal among all compression methods.

3. Result and Discussion

The design of standard H.264 Video Encoder is done using Verilog and implemented in a Xilinx Spartan 3E XC3S500E (package: fg320, speed grade: -4) FPGA using the Xilinx ISE 14.1i design tool The internal RTL schematic of the Encoder is shown in fig.1.

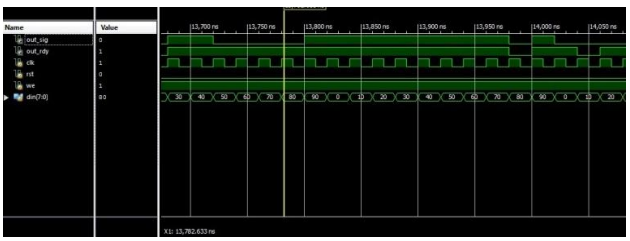

Fig. H.264 Video Encoder Output

DCT is modified using Strassen Matrix. The Strassen matrix multiplier as compared to standard matrix multiplier shows much more reduction in device Utilization and delay. The standard H.264 Encoder and Modified Encoder is implemented in Spartan 3 fpga and the output is shown in the figure.

Conclusion

H.264 Video Encoding Standard is presented and implemented in FPGA based on utilizing Verilog. The design is implemented on Xilinx Spartan 3 XC3S700AN FPGA device. The conventional DCT is modified using Strassen Matrix multiplier. The aim is to present a comparative study of the standard H.264 Video Encoding Standard and Modified H.264 Video Encoding Standard. The Strassen matrix multiplier as compared to standard matrix multiplier shows much more reduction in device Utilization and delay. The delay of standard 8x8 multiplier is 32.05ns and that to Strassen 8x8 matrix multiplier delay is only 23.602ns. The Strassen 8x8 matrix multiplier uses only 7044 slices out of 92152 slices. Strassen matrix multiplication provides an efficient method for reducing the delay and area of matrix multipliers.

References

Thomas Wiegand, Gary J. Sullivan, Overview of the H.264/AVC Video Coding Standard, IEEE transactions on circuits and systems for video technology.

S. Aslan, E. Oruklu and J. Saniie, Realization of Area Efficient QR Factorization using Unified Division, Square Root and Inverse Square Root Hardware, in IEEE Electro/Information Technolog

D. Jeffrey and W. Zhou, Fraction-free matrix factors: new forms for LU and QR factors, Frontiers of Computer Science in China, Springer-Verlag Gm

J. A. Apolinario Jr, QRD-RLS Adaptive Filtering. New Jersey: Prentice H

B. Kakaradov, Ultra-fast Matrix Multiplication: An Empirical Analysis of Highly Optimized Vector Algorithm,Stanford Undergraduate Research Journa

S. Choi, V.K.K. Prasanna and J. Jang, Area and time efficient implementations of matrix multiplication on FPGAs, in IEEE International

William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery, Numerical Recipes 3rd Edition: The Art of Scientific Computing, rd ed. New York City: Cambridge University Press

D. Takahashi, T. Boku, M. Sato and Y. Ohtaki, Parallel implementation of Strassens matrix multiplication algorithm for heterogeneous clusters,in parallel and Distributed Processing Symposium

S. Robinson, Toward an Optimal lgorithm for Matrix Multiplication, SIAM News

A. Gerstlauer, R. Domer, D.D. Gajski and S. Dongwan, An Interactive Design Environment for C-Based High-Level Synthesis of RTL Processors, IEEE Transactions on Very Large Scale Integration (VLSI) Syste

A. Morawiec and P. Coussy, High-Level Synthesis: from Algorithm to Digital Circuits. Berlin: Springer Science + Business Media.