LOW POWER LOSSSLESS COMPRESSION OF REAL TIME MPEG4 VIDEO ENCODING AND DECODING USING VHDL AND MATLAB

Aswathy Prasad, Kamya Krishnan, Karthika, Parvathy, Rajesh Kannan Megalingam,
Department of Electronics and Communication
Amrita Vishwa Vidyapeetham, Kollam, Kerala, India
aswathyprasad87@gmail.com, kkris59@gmail.com, karthika06031987@gmail.com, orchidkollam@gmail.com, rajeshm@amritapuri.amrita.edu

Abstract: - This paper presents a simple model of complex real time MPEG-4 video encoding and decoding using simple techniques in VHDL and MATLAB to provide reasonable compression while using only less power and resources on FPGA. This implementation works on low power and less number of clock cycles. The basic video codec module consists of a video encoder and a decoder. The encoder module consists of blocks for temporal modeling, spatial modeling and Entropy encoding. Temporal block has a difference block whereas spatial block consists of 2-D DCT, Quantizer and a 2-D IDCT. Sample frames of real time video has been processed using the codec module resulting in an average compression of 64.6% It can be applied in areas where low bit rate, high quality video is required. The first 5 sections in this paper represents the concepts and theories used, section 6 onwards represents the actual implementation of the module.

Index Terms:-MPEG-4 video codec, Temporal modeling, Spatial modeling, Entropy encoding, Discrete Cosine Transform, VLC.

I. INTRODUCTION

Network bitrates continue to increase (dramatically in the local area and somewhat less so in the wider area), high bit rate connections to the home are commonplace and the storage capacity of hard disks, flash memories and optical media is greater than ever before. With the price per transmitted or stored bit continually falling, it is perhaps not immediately obvious why video compression is necessary (and why there is such a significant effort to make it better.) [2]

Video compression has two important benefits. First, it makes it possible to use digital video in transmission and storage environments that would not support uncompressed ('raw') video. Second, video compression enables more efficient use of transmission and storage resources[2]. There are two basic issues to be addressed in the design of codec for video applications: speed of processing and power considerations. With the proliferation of personal computers in multimedia applications and the evolution of digital networks, speed of processing is the most vital need for effective real time communication. Increase of clock frequency, operating voltage or system complexity increases the power consumption drastically. The power consumed in integrated circuits is a significant and increasing portion of the total system power consumption. Thus, the developments of low power VLSI design methodologies are inevitable [6] [7].

II. THE MPEG STANDARD

The MPEG-4 is a coding standard that supports ways for communication, access and manipulations of audio and video data. In particular the MPEG-4 is designed to manipulate the audio-visual information in terms of macro blocks. It is developed from the MPEG video group by providing standardized core technologies allowing efficient storage, transmission and manipulation of video data in multimedia environments. [1][2].

MPEG-4 Visual provides a highly flexible toolkit of coding techniques and resources, making it possible to deal with a wide range of types of visual data including rectangular frames ('traditional' video material), video objects (arbitrary-shaped regions of a visual scene), still images and hybrids of natural (real-world) and synthetic (computer-generated) visual information. MPEG-4 Visual provides its functionality through a set of coding tools, organised into 'profiles', recommended groupings of tools suitable for certain applications. Classes of profiles include 'simple' profiles (coding of rectangular video frames), object-based profiles (coding of arbitrary-shaped visual objects), still texture profiles (coding of still images or 'texture'), scalable profiles (coding at multiple resolutions or quality levels) and studio profiles (coding for high-quality studio applications). [4]

III.MOTION COMPENSATION AND ESTIMATION

The difference between consecutive frames is small in a video sequence. This feature can therefore be exploited to get better compression. If the encoder discovers that a part *P* of the preceding frame has been rigidly moved to a different location in the current frame, then *P* can be compressed by writing the following three items on the compressed stream: it's previous location, its current location, and information identifying the boundaries of *P*. The encoder scans the current frame block by block. The encoder writes the difference between its past and present locations on the output. This difference is of the form

$$(Cx - Bx, Cy - By) = (\Delta x, \Delta y),$$

so it is called a *motion vector*. In general, motion compensation is lossy [1][3][6].

Motion estimation involves finding a sample region in a reference frame that closely matches the current block. The reference frame is a previously encoded frame from the sequence and may be before or after the current frame in display order. An area in the reference frame centered on the current block position (the search area) is searched and the region within the search area that minimizes a matching criterion is chosen as the 'best match '[1][3][6].

IV. SOURCE ENCODING

Few of the Variable Length Codes (VLC) which is used to implement the source encoder is Arithmetic code, Golomb rice code and Huffman code. Due to high complexity of Arithmetic coding and decreased compression rate of Golomb Rice coding, Huffman code is chosen for source encoder which is implemented in MATLAB.

The Huffman code is a source code whose average word length approaches the fundamental limit set by the source entropy H(s).Huffman code uses the statistics of the source to generate a binary sequence that represents the source symbols. The Huffman Algorithm proceeds as follows:

1. The source symbols are listed in order of decreasing probability. The two source symbols with the lowest probability are assigned a 0 and a 1 (Splitting).
2. The two source symbols are regarded as being combined into a new source symbol with probability equal to the sum of the two original probabilities. The probability of the new symbol is placed in the list in accordance with its value.
3. The procedure is repeated until we are left with a final list of source statistics of only two for which a 0 and a 1 is assigned.

The code for each original source symbol is found by working back and tracing the sequence of 0s and 1s assigned to that symbol as well as its successors [5].

V. DISCRETE COSINE TRANSFORM

The purpose of the transform stage in an image or video CODEC is to convert image or motion-compensated residual data into frequency domain (transform domain). The Discrete Cosine Transform (DCT) operates on X, a block of $N \times N$ samples (typically image samples or residual values after prediction) and creates Y, an $N \times N$ block of coefficients. The action of the DCT (and its inverse, the IDCT) can be described in terms of a transform matrix A. The forward DCT (FDCT) of an $N \times N$ sample block is given by

$$\mathbf{Y} = \mathbf{AXA}^{\mathrm{T}}$$

and the inverse DCT is given by,

$$\mathbf{X} = \mathbf{A}^{\mathrm{T}}\mathbf{YA}$$

where X is a matrix of samples, Y is a matrix of coefficients and **A** is an $N \times N$ transform matrix. The elements of A are:

$$A_{ij} = C_i \cos \frac{(2j+1)i\pi}{2N}$$

$$C_i = \sqrt{\frac{1}{N}}\ (i = 0), \qquad C_i = \sqrt{\frac{2}{N}}\ (i > 0)$$

DCT and IDCT in summation form:

$$Y_{xy} = C_x C_y \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} X_{ij} \cos \frac{(2j+1)y\pi}{2N} \cos \frac{(2i+1)x\pi}{2N}$$

$$X_{ij} = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} C_x C_y Y_{xy} \cos \frac{(2j+1)y\pi}{2N} \cos \frac{(2i+1)x\pi}{2N}$$

On performing DCT in an image block, the information content of the image is concentrated on the lower frequency components of the transform. Setting all the coefficients to zero except the most significant (coefficient 0, 0, the 'DC' coefficient) and performing the IDCT gives the output block which is very much similar to the original image. Hence it is possible to reconstruct an approximate copy of the block from a subset of the DCT coefficients. Removing the coefficients with insignificant magnitudes enables image data to be represented with a reduced number of coefficient values at the expense of some loss of quality[1][3].

Now we present a simple implementation MODEL for a highly complex video processing using VHDL and MATLAB coding which uses less clock cycles and power.

VI. CODEC MODULE

Data compression be it audio, video or image is mainly achieved by removing the redundancy in the input data and source encoding of the residual data. Since video has huge amount of data to process it will intuitively take more power. Video data shows redundancy in two major aspects, they are temporal redundancy and spatial redundancy. (Fig 1)

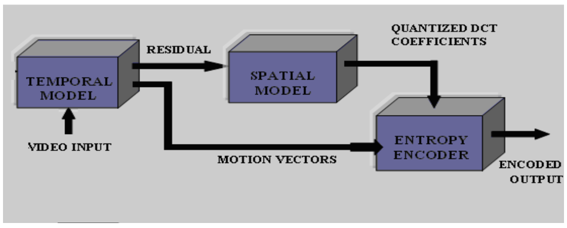

Fig 1. Video encoder block diagram

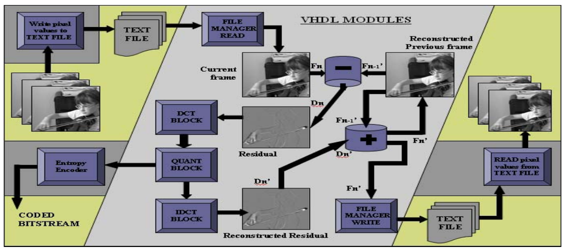

Fig 2. Internal block diagram of video encoder

VII. TEMPORAL MODULE

Temporal redundancy refers to the large amount of similar data present in adjacent frames of any real time video. The input and output to the temporal model is shown in Figure 3. It takes previous reconstructed frame and current frame as input and outputs a residue frame which is the difference between the two input frames. As is seen from Fig.2 and Fig 3, the residual frame is prominent only at places where motion has been encountered. Thus, most of the values in the residual frame are zeroes except for those places where movement has taken place. Temporal block consists of difference block, adder bock and motion compensation and motion estimation blocks. As a simple model for ENCODER is being implemented, the most significant module of TEMPORAL MODELLING, the difference block has been the target of study. It was simulated and tested in MODELSIM and then synthesized in both SPARTAN3-XC3S400 and a high performance VIRTEX5.
If the input frames are of 256x256 sizes, difference block has to find the difference of 65536(256*256) pixel values. It takes input from FILE MANAGER BLOCK and ADDER BLOCK. The maximum value that can go as input to the block is 255 (corresponding to black pixel). Hence 16(two 8 bit lines each from FILE MANAGER BLOCK and ADDER BLOCK) bit lines are required as input. Two flag bits are also coming as input from these blocks. The output is 9 bits, MSB for sign bit and a flag bit to enable the subsequent block, DCT. Difference block takes in the two inputs from FILE MANAGER and ADDER simultaneously and compares them to find the largest number. Then the largest is subtracted form the smallest with sign bit added as MSB and goes as output.

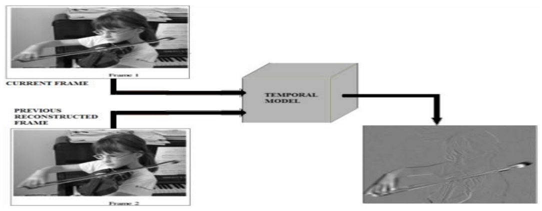

Fig 3. Temporal model output.

HDL Synthesis Report on SPARTAN3-XC3S400

8-bit subtractor	2
1-bit register	18
8-bit register	2
1-bit xor2	8

Device utilization summary:-

No: of Slices:	29 out of 3584	0%
No: of Slice Flip Flops:	33 out of 7168	0%
No:of 4 input LUTs:	45 out of 7168	0%
No: of bondedIOBs:	30 out of 141	21%
No: of GCLKs:	1 out of 8	12%

The maximum frequency of the circuit is 189.581 MHz. The total power for the block was found out and is showing the conventional trend of increase with increase in input frequency. (Fig 4) The results are shown below.

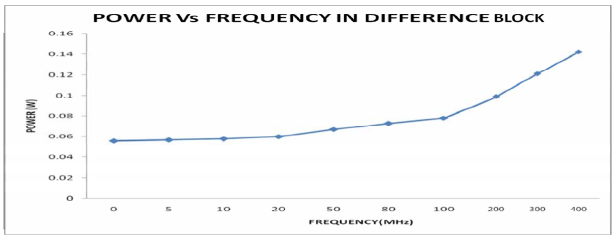

Fig 4: Power vs Frequency for difference block.

VIII. SPATIAL MODULE

Spatial redundancy refers to the similarities of the pixels within a frame. In any frame in a video, the adjacent pixels are almost the same. This redundancy is removed using the spatial model (Fig.2, Fig 5), and hence the second stage of video compression is achieved in the spatial block. By using Discrete Cosine Transform (DCT) the pixel values are transformed into frequency domain in which most of the energy of the transformed frame is concentrated on the lower frequency components. On further quantizing the obtained DCT coefficients, the resulting values can be represented in lower number of bits. The internal block diagram of the spatial model is shown in Fig. (5). The DCT calculation is done in DCT block followed by quantizing the DCT coefficients in the Quantization block. Quantization thus results in lower range of values which can be represented using lower number of bits. QUANTISATION BLOCK thus converts all the real valued DCT coefficients to integer values, thus producing perfect input for source encoding/ channel encoding of the quantized transform coefficients.

The block has 17 bit input corresponding to the integer portion of DCT coefficient, and 1 bit corresponding to the decimal portion of it. The QUANTIZER has 17 bit and a flag bit as output to enable the subsequent block, IDCT and source encoder. Therefore, it has a total of 39 bit lines going into and from the block.

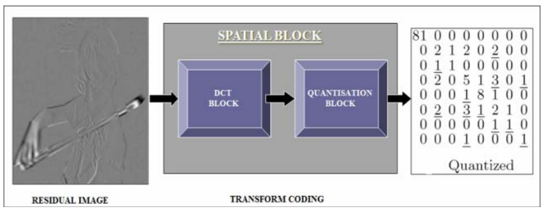

Fig 5. Spatial model Output.

HDL Synthesis Report on SPARTAN3-XC3S400

16 bit adder	1
1 bit register	18

Device utilization summary:-

No: of Slices:	18 out of 3584	0%
No: of Slice Flip Flops:	33 out of 7168	0%
No: of 4 input LUTs:	45 out of 7168	0%
No: of bondedIOBs:	39 out of 141	27%
No: of GCLKs:	1 out of 8	12%

The total power for the block was found out and is showing the conventional trend of increase with increase in input frequency. The results are shown below.

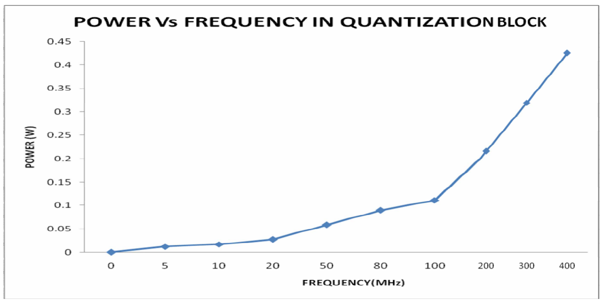

Fig 6: Power vs Frequency for Quantization block

IX. VIDEO DECODER

The compressed coded bit stream is then passed through Huffman decoder where the quantized coefficients are reconstructed back and is given to the IDCT block. Using these quantized coefficients, the IDCT block transforms it back to the original domain to get the residual frame back. This residue is added with the previous frame already stored in the adder block to obtain the current frame. The previous frame stored in the adder block is now replaced with current frame for the future frames. (Fig 7)

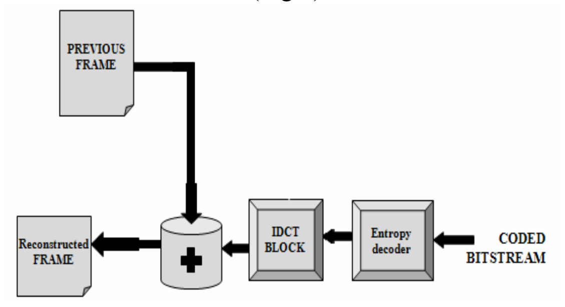

Fig 7. Internal block diagram of video decoder

X. CLOCK REQUIREMENTS OF THE BLOCKS.

One clock is required for the DIFFERENCE BLOCK to calculate the difference between the pixel values. Since a block of 8x8 is taken, 1 macroblock wil need 64 clock cycles to get the residue. Once the residue is obtained one DCT coefficient is calculated in one clock cycle thus getting all the 64 coefficients in just 64 clocks. As the first DCT value is calculated QUANTIZER starts and then both DCT and QUANTIZER works parallely. When all 64 DCT values are obtained IDCT block begin to reconstruct the pixel values. We get one pixel value in one clock. And simultaneously the values are stored in the ADDER block for subsequent prediction.(Fig 8)

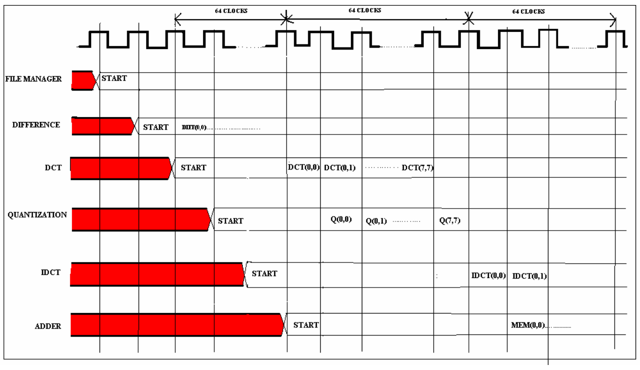

Fig 8: Clock utilization of each block in encoder

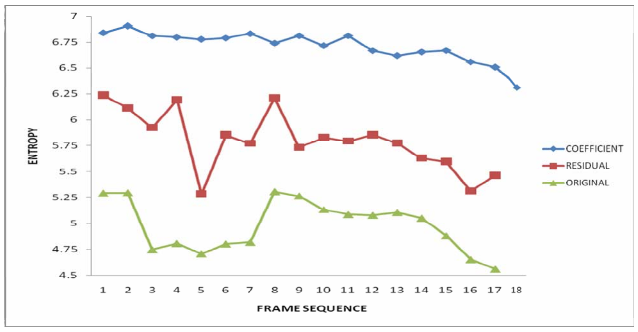

Fig 9: Entropy of original frames, residues and DCT Coefficients

XI. RESULT OBTAINED

The main experiment that we had conducted was extracting a 0.25 sec video sequence consisting of 6 frames of size 256 x 256 pixels from a real time MPEG-4 video sequence and computing the power for a temporal block module namely DIFFERENCE block and a spatial block module namely QUANTIZER. The six frames were then given as input to the MATLAB cum VHDL module for the process of Video Encoding. The time required to process each frame was 7 ms.

After passing the input frames through DIFFERENCE block a compression of 19% was obtained. And after Spatial modeling, compression increased by 28.6%. So a total of 44.6% compression was achieved by using DIFFERENCE block alone rather than motion estimation and motion compensation blocks. (Fig 9) But on further compressing it by entropy encoding, additional compression of 20% was obtained resulting in a net compression of 64.6% without much quality loss in the video. Compression was calculated using Shannon's Entropy Theorem.

Motion estimation and compensation blocks take 66% of power in a codec module and the compression is lossy. But Difference block which also gives reasonable compression takes only less than 10% of the total power with lossless compression. (Fig 9). Hence this design has an acceptable trade off between power and compression without compromising on the speed of processing.

We hence implemented a basic technique for the encoding and decoding of the highly complex real time MPEG-4 video processing and computed the power requirements and compression obtained. Tools used were MODELSIM for simulating the VHDL modules, XILINX 10.1 for synthesis, and MATLAB for implementing complex VLC part for source encoding.

XII. CONCLUSION

Video compression algorithms operate by removing redundancy in the temporal, spatial and/or frequency domains. Most practical video compression techniques are based on lossy compression, in which greater compression is achieved with the penalty that the decoded signal is not identical to the original. The goal of a video compression algorithm is to achieve efficient compression whilst minimizing the distortion introduced by the compression process. Here reasonable compression is achieved along with very less distortion in the quality of video and without using as much power as original encoder. Even with constant advances in storage and transmission capacity, compression is likely to be an essential component of multimedia services for many years to come.

ACKNOWLEDGEMENT

We gratefully acknowledge the Almighty GOD who gave us strength and health to successfully complete this venture. The authors wish to thank Amrita Vishwa Vidyapeetham, in particular the Digital library, for access to their research facilities.

REFERENCES

[1] H.264 and MPEG-4 Video Compression by Iain E.G. Richardson.
[2] Data Compression by David Salmon.
[3 Image compression and the Discrete Cosine Transform by Ken Cabeen and Peter Gent
[4] MPEG 4 Video Encoder Based on DSP-FPGA Techniques by Jianwei Niu, Rui He, Jianping Hu.
[5] Digital Communication by Simon Haykins
[6] Algorithms , Complexity analysis and VLSI Architectures For MPEG4 Motion by Peter M Kuhn
[7 Digital VLSI Systems Design –A Design Manual For Implementation of projects by Seetharaman Ramachandran.
[8] Digital Image Processing by William K Pratt.
[9] Digital Video Compression – Paper by Joseph B Walt rich
[10] New Parallel Architecture for the DCT and Its Inverse for image compression – Paper by Bousselmi M, Bouhlel M S, Masmoudi N, Kaumun
[11] Image Compression and the Discrete Cosine Transform- Paper by Ken Cabeen and Peter Kant.
[12] MPEG 4 Video Encoder Based On DSP-FPGA Techniques by Jianwei Niu, Rui Hei, Jianping Hu.
[13] Memory Efficient Design of A MPEG 4 Encoder For FPGA – Paper by Kristof Denolf, Adrian Chirila Rus, Robert Turney, Paul Schumacker, Kees Vissers.
[14] Power Aware FPGA Design by Hischem Belhadj, Vishal Aggarwal, Ajay Pradhan, Amal Zerrouki.
[15] JPEG, MPEG4, H.264 Codec IP Development-Paper by Chung Jr Lian, Yu Wen Hun, Hung Chi Fang, Yung Chi Chang, Liang Gee Cheng.