Design and FPGA Implementation of Fast Variable Length Coder for a Video Encoder

N. Venugopal * and Dr S. Ramachandran**

* M. G. R. University, Chennai, India
** National Academy of Excellence, Bangalore, 560-060, India

Summary
This paper proposes a novel implementation of one of the core processors of a video encoder, the variable length coder using single FPGA. The processor is implemented on a Xilinx Virtex – II Pro XUPVP30 FPGA. The gate count of the implementation is approximately 690,000 including an output FIFO of size 128 Kb. It can process 1600x1200 pixels color motion pictures in 4:2:0 format at over 30 frames per second as per MPEG-2 standard. The compression effected is about 38 and the reconstructed picture is of good quality with a PSNR values of 33 dB or more.

Key words:
DCTQ, Encoder, RLE, FIFO, VLC.

1. Introduction

Video compression coding is the enabling technology behind a new wave of communication applications. From streaming Internet video to broadcast digital television and digital cinema, the video codec is a key building block for a host of new multimedia applications and services. Certain applications, such as medical diagnostics, space exploration etc. requires high resolution image/video at high frame rate which necessitates huge bandwidths and storage space. For example, one hour of color motion picture of frame size 1024x768 pixels at 25 frames per second in the raw format will require about 210 GB of memory and 470 Mbps channel speed for effective communication. Video Compression is, therefore, essential to bring down the storage requirements to manageable levels and to transmit data with the existing channel capacities.

Moving Pictures Experts Group (MPEG) is a compression standard group established by the ITU and ISO. MPEG is a generic means of compactly representing digital image, video and audio signals. The MPEG-1 standard, established in 1992, is designed to produce reasonable quality images and sound at low bit rates at 1.5 M bits/sec. The MPEG-2 standard [1] is designed to produce broadcast quality video at higher bit rates up to 100 Mbps.

Over the years, many attempts have been made to implement different functional modules of the Video Codec in VLSI [2-6]. H. Park et al. [3] and H.C. Chang et al. [4] have proposed VLSI architectures for Variable Length Coder (VLC) for MPEG-1 and JPEG respectively with limited throughputs and hence not suitable for high resolution motion pictures. One of the present authors developed earlier a fast algorithm for the DCTQ and implemented the same on FPGA [5]. This module is used in the present implementation for generating the DCTQ coefficients for testing a video sequence. The same author had also developed VLC module [6], which was slow in comparison with the DCTQ processor by about 40%. As an ongoing project for space application for processing very high resolution color pictures in the order of 1600x1200 pixels, the proposed VLC Processor has been completely re-designed resulting in higher throughput by at least 30% over Ref. [6]. In the recent years, few more VLC/VLD implementations have been reported [7-11].

This paper is organized as follows: The next section presents the basic architecture of video encoder followed by the detailed architecture of the VLC and RLE implemented and the principles involved therein. Results are discussed in Section 4 and Conclusions are presented in Section 5.

2. Architecture of Video Encoder

A video sequence that needs to be compressed is applied at the input of a Discrete Cosine Transform (DCT) module as shown in Fig. 1. The DCT prepares the ground for compression by packing the picture information in as few coefficients as possible. This is followed by quantization (Q), which succeeds in converting the DCT coefficients into many zeros which need not be coded in the next stage, the variable length coder. The VLC assigns shortest possible codes of varying lengths to the DCTQ coefficients, thus bringing about compression of the motion picture. The compression effected in VLC is much greater than that achieved during quantization. The VLC compression is lossless, whereas quantization is lossy. A brief description of various modules of the video encoder as implemented by one of the authors is as follows.

Manuscript received July 5, 2009
Manuscript revised July 20, 2009

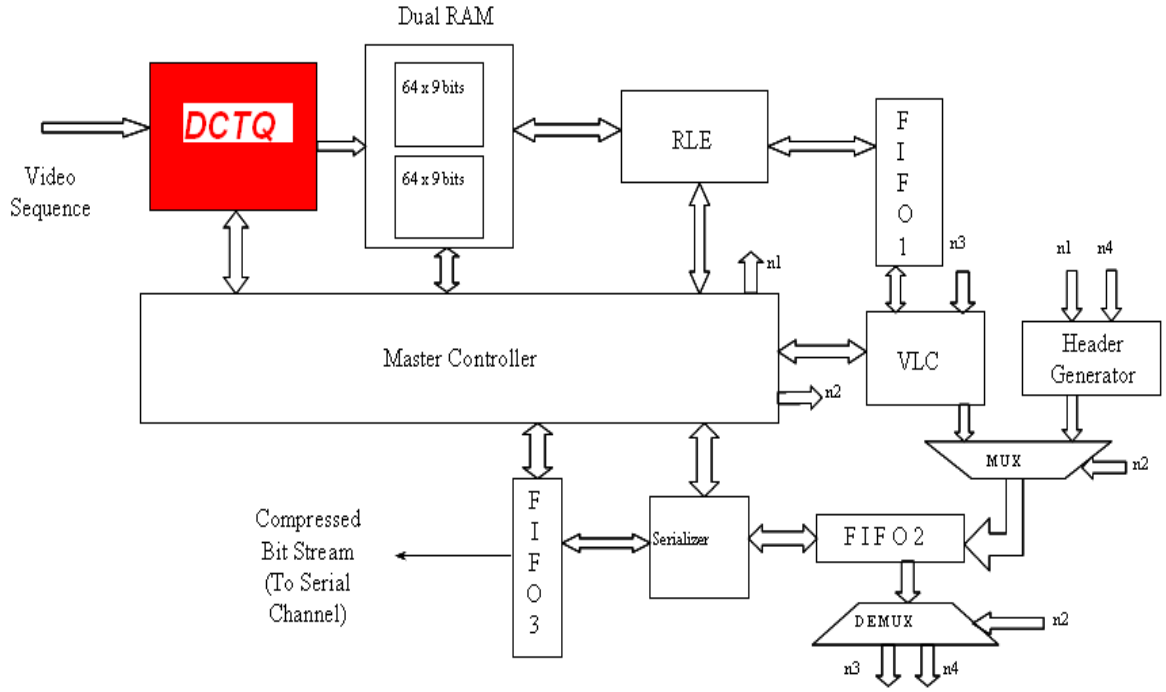


Figure 1 Architecture of Variable Length Coder of Video Encoder Implemented

2.1 Discrete Cosine Transform

A 2-D DCT is performed on 8x8 pixel blocks. The magnitude of DCT coefficient signifies the combination of horizontal and vertical spatial frequencies to the original picture block. In most natural pictures, most of the energy is concentrated in the DC and low frequency terms. Thus these coefficients are much higher in magnitude than the other coefficients and the subsequent quantization reduces most of the higher frequency terms to zero.

2.2 Quantization

Quantization forms the lossy part of the MPEG compression. Most of the high frequency terms are rounded to zero upon quantization. Also the degree of quantization for the higher frequency terms is more as compared to that of low frequency terms. Thus, quantization reduces the possible values to be transmitted and leads to compression.

2.3 Run Level Encoding and DC Differential

Run Level Encoding (RLE) exploits the presence of many zeros in the DCTQ output by scanning the DCTQ output in a zigzag manner and thus reducing the physical size of repeating strings of zeros. The numbers of zeros traversed before reaching a non-zero AC coefficient is called the Run length and the AC coefficient is called the

Level. Thus, the AC coefficients are encoded as Run-Level while the DC term is coded as differential DC.

2.4 Variable Length Code

The list of values produced by scanning (RLE) is entropy coded using a variable length code. Each VLC code word denotes a run of zeros followed by a non- zero coefficient of a particular level. The differential DC is also encoded as variable length code.

2.5 Buffer

A consequence of using different picture types and variable length coding is that the overall data rate is variable. In applications that involve a fixed-rate channel, a FIFO buffer may be used to match the encoder output to the channel. This was also developed in the present work. In the present work, all the modules except DCTQ module shown in Fig. 1 have been developed. Detailed descriptions of the designed modules are as follows.

3. Run Length Encoding

In this section, the fundamentals of DC differential and Run Length Encoding are discussed. The RLE module operates on the DCTQ output to calculate the DC differential and the Run-Level combinations. RLE reads its input from a Dual RAM as shown in Fig. 1 and writes its output in a FIFO1. The Verilog HDL implementation of the RLE module is discussed in the next sub section.

The coding of the quantized DCT coefficients exploits the likely clustering of energy into the low-frequency coefficients and frequent occurrence of zero-value coefficients. The block is scanned in a diagonal zigzag scanning pattern starting at the DC coefficient to produce a list of quantized coefficient values, ordered according to the zigzag scan pattern. The number of zeros is counted before a non-zero AC coefficient is reached. The count of zeros is called run length and the non-zero AC coefficient is called the level. For example, consider the quantized DCT output shown in Table 1. The run length encoding in zigzag order for the above mentioned quantized coefficients is shown alongside.

Apart from run-length encoding, the RLE module also calculates the differential DC. To calculate differential DC, a predictor is subtracted from the input DC term. For this purpose, three predictors are maintained, one each for the 3 color components (Y, Cb and Cr). The predictor is set to the value equal to the last DC term encoded. To begin with, the predictors are reset to a value of 256 since the precision implemented in the present work is 9 bits for DCTQ coefficients as per MPEG-2 standard. The predictors are also reset at the beginning of each slice.

Table 1 A Sample Block of DCTQ Coefficients and its Run Length Encoding

130	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
-4	0	10	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Run Length	Level
1	3
0	2
6	-4
12	10
EOB	

3.1 Architecture of Run Length Encoder

RLE processor reads from the Dual RAM shown in Fig. 1, in which the DCTQ coefficients are stored by the DCTQ processor. Dual RAM serves as double buffer, thus maintaining continuous processing of RLE. The RLE processor writes the Run length and Level outputs computed (such as the sample block presented in Table 1) into FIFO1. The RLE processes the input data and stores the output in FIFO1 only if an empty location is available. Otherwise, RLE automatically holds its operation.

The Architecture of the Run Length Encoder is shown in Fig. 2. The RLE is reset at the time of power on condition by asserting the asynchronous, active low signal “reset_n”. RLE operation can be activated by asserting “enable” signal. “sys_clk” serves as the system clock. The number of slices in the input frame is specified by the input “slice_count [5:0]”. Similarly the “mb_count [5:0]” specifies the number of macroblocks in one slice for the input frame. The signal “wfull_fifo_rle” indicates whether the FIFO1 is full or not. The “alternate_scan” specifies the type of zigzag scan as per the standard. The signal “chroma_format [1:0]” selects one of the macroblock formats 4:4:4, 4:2:2 or 4:2:0.

The DCTQ output is input at the pins marked “zigzag_in [8:0]”. The color component (Y, Cb or Cr) that is to be processed is specified in “chroma_format [1:0]”. The RLE output is issued at the output pins “rl_out [14:0]”. The “rl_out [14:9]” contains the run length while the “rl_out [8:0]” contains the level. The differential DC is also output at the same pins “rl_out [11:0]” with “rl_out [14:12]” forced to ‘000’. The signal “re_addr [5:0]” serves as the read address of the Dual RAM in which DCTQ coefficients are stored earlier. The signal “winc_fifo_rle” serves as the write enable for the FIFO1, in which RLE writes its outputs. The signal marked “frame_done” indicates that the encoder has finished processing a video frame. Immediately after one block of DCTQ coefficients is processed, the RLE starts processing the next block stored in the Dual RAM.

3.2 Variable Length Coding

In this section, the implementation details of Variable Length Coder, Header Generator and Serializer are presented. Like previous modules, these modules were also implemented in Verilog HDL. The Run Level Encoder (RLE) writes its output in the FIFO1. In a FIFO, data are written sequentially using one clock domain, while the data are read sequentially from the same FIFO using another clock domain; the two clock domains being asynchronous to each other. The FIFO is a dual-port RAM addressed by Gray counters. The FIFO full or empty status is generated by asynchronous comparison of the read and write pointers and is set asynchronously. This FIFO design of Xilinx is adopted in the present work.

The full/empty status of the FIFO decides whether the write/read to the FIFO should hold its operation or not. Thus, the use of FIFO eliminates the handshaking between the read and the write operations.

FIFO helps in transferring data between different clock domains. Thus the VLC clock, the Serializer clock and the

output clock can be set to the desired value without any need for inter-clock domain synchronization. FIFO1 is of depth 8 and width 15 bits (RLE output), FIFO2 is of depth 4 and width 29 bits (VLC output) and FIFO3 is of depth 128 K x 1-bit (width of output of Serializer).

3.3 Encoding DC Differential

DC differential (for each DC coefficient) of a block in intra macroblocks is encoded as a variable length code. If it is equal to zero, then it is followed by a fixed length code.

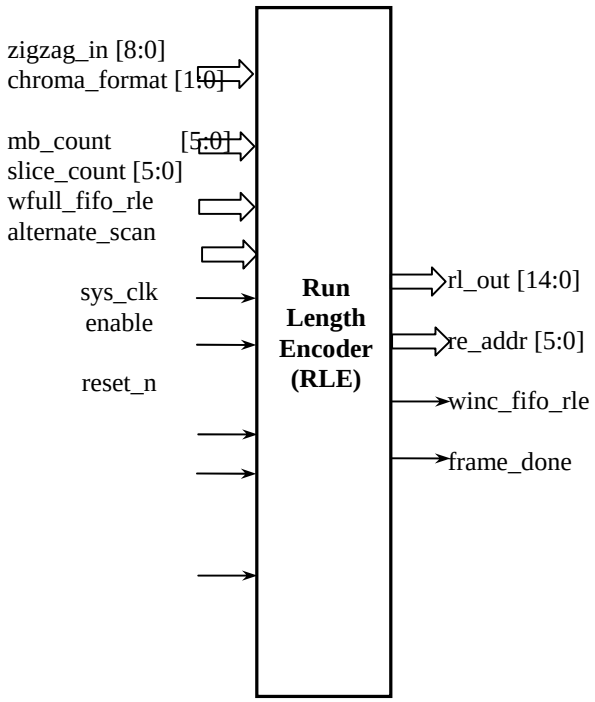


Figure 2 Architecture of Run Length Encoder

Otherwise, it is assigned variable codes as defined in D.12 Table of MPEG-2 standard. At the decoder end, a differential value is first recovered from the coded data, which is added to the predictor in order to recover the DC coefficient. For example:
DC coefficient = 130 (Luminance) and Predictor = 128
DC differential = DC coefficient – Predictor = 2
From Table D.12 of the standard, the code is “01” for the luminance, followed by its binary value of “10” Thus, the VLC code for this DC coefficient is 01 10.

3.4 Encoding Run Level

The list of values produced by zigzag scanning (RLE) is entropy coded using a variable length code (VLC). Each VLC code word denotes a run of zeros followed by a non-zero AC coefficient. VLC coding recognizes that short runs of zeros are more likely than long ones and small coefficients are more likely than large ones.

Table 2 Variable Length Code Assigned

Run Length	Level	VLC Code
1	3	0010 0101 0
0	2	0100 0
6	-4	0000 0100 0110 1111 1100
12	10	0000 0100 1100 0000 1010
EOB		10

The VLC allocates code words, which have different lengths depending upon the probability with which they are expected to occur. To enable the decoder to distinguish where one code ends and the next begins; the VLC has the property that no complete code is a prefix of any other. The list of run lengths and levels is coded using Table B-14 given in the Ref. [1]. Run is a 6-bit fixed length code and Signed_level is a 12-bit fixed length code. For example, the VLC codes for the run level presented in Table 1 are shown in Table 2.

3.5 Architecture of VLC

VLC reads the DC differential and run-level from a FIFO (FIFO1 in Fig. 1) in which the RLE module writes its output. Thereafter VLC processes (as explained in Section 2.2) and writes the computed variable lengths for the DC differential as well as the AC coefficients into another FIFO marked FIFO2 in Fig. 1. The VLC takes an input from FIFO1 only if it is not empty, processes it and writes the output in FIFO2 if any empty location is available in FIFO2. Otherwise, VLC automatically holds the operation. As in the RLE, the VLC processor is reset at the time of power on condition by asserting the asynchronous, active low signal “reset_n”. Similarly, the VLC operation can be activated by asserting “enable” signal. “sys_clk” serves as the system clock. The input signal “slice_count [5:0]” specifies the no. of slices in the input frame. The number of macroblocks to be processed in one slice is specified in “mb_count [5:0]”. The input “wfull_fifo_vlc” indicates whether the FIFO2 in which the VLC has to write its output is full or not. The input “rempty_fifo_rle” specifies whether the FIFO1 from which VLC has to read its input is empty or not. “chroma_format [1:0]” specifies the macroblock structure as one of 4:4:4, 4:2:2 and 4:2:0. Inputs “rl_in [14:9]” contains the run and “rl_in [8:0]” the level computed by the RLE processor. The input “quantiser_scale_code [4:0]” is required as a part of the slice header. The output “vl_size_code [28:0]” contains the desired VLC code for a particular run-level or DC differential. The code size is available in bits [28:24] and the actual VLC code in bits [23:0]. “rinc_fifo_rle” is the read enable output for the FIFO1 from which VLC reads its input. The output signal “winc_fifo_vlc” is the write

enable for the FIFO2 in which VLC writes its output. The output "frame_done" signals when the VLC has finished processing a frame.

Apart from variable length code generation for differential DC and run-level, VLC module also outputs a part of header (slice header, macroblock address increment and macroblock modes). At the beginning of each frame, the Header Generator embedded inside the VLC module outputs the Sequence Header, Sequence Extension Header, Picture Header and Picture Coding Extension Header. It outputs the headers in chunks of 24-bits. The Header and VLC output are multiplexed (by the MUX shown in Fig. 1) and the master controller issues the appropriate select signals as shown.

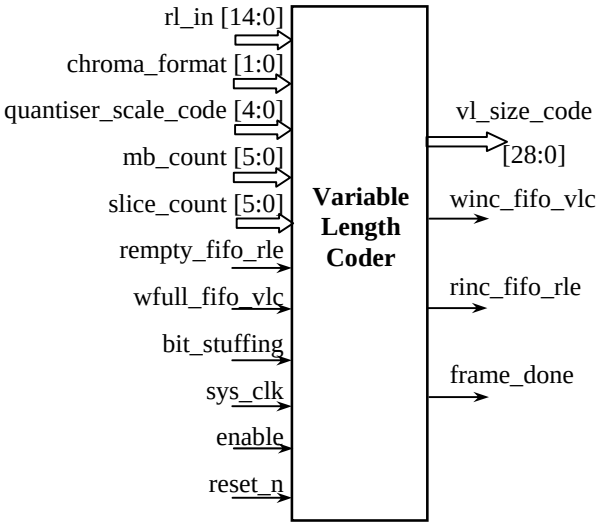


Figure 3 Architecture of Variable Length Coder

When the frame processing starts, the Header Generator outputs the header information. Once the "header_done" signal goes high, the VLC output is selected till the end of processing of that frame. Similarly the FIFO2 full/empty status is either sent to Header Generator or the VLC.

4. Results and Discussions

In the present work, the entire Variable Length Coder presented in Fig. 1 has been coded in Verilog conforming to RTL coding guidelines. Modelsim of Mentor Graphics has been used to verify the functionality of the VLC design. The simulation results for the RLE and VLC processors are as follows.

4.1 Simulation Results

RLE reads from the Dual RAM and writes its output into FIFO1 as presented in the RLE processor simulated waveform in Fig. 4. The signal "zigzag_in" is the input to the RLE module. Whenever RLE writes its output into the FIFO1, it generates a pulse "winc" (write_enable for the FIFO) if the "wfull" (Full status signal of FIFO) is not high. It may be noted that in this snapshot; once when the "wfull" is high, the RLE module stays in its state 6 till the time "wfull" goes low.

The simulated waveform for the VLC processor is shown in Fig. 5. VLC reads its input (rl_in) from FIFO1 and write its output (vl_size_code) in FIFO2. It may be noticed that after every "rinc" pulse, VLC gets a new input and after the code is generated, the VLC module generates a "winc" pulse in order to write the output (if the "wfull" is not high). The signal "ba_bits" represents the number of bits required for byte alignment. In the present work, the number of clock cycles required to process a block (assuming "rempty" and "wfull" to be low) are 1 clock cycle to read DC differential plus 3 clock cycles to compute the DC differential code plus 1 clock cycle to write the DC differential code plus 3 x (the number of run-level entries for a block) plus 1 clock cycle to read EOB plus 1 clock cycle to write EOB plus 3 clock cycles to update blocks processed. That is, total time required for processing Macroblocks and Slices may be expressed as 10+3n, where 3 includes reading, computing VLC code and writing the code and, n is the number of run-level entries for the block. The simulation results presented in Fig. 6 for a sample picture shows good quality of reconstructed picture (33 dB) and compression achieved being 38. Similar results were obtained for many video sequences that were experimented with.

4.2 Synthesis and Place & Route Results

The various modules of the VLC Processor were synthesized using Synplify Pro 7.7.1 and place and routed using Xilinx Project Naviator. The target device chosen was Xilinx Virtex-II Pro XUPVP30 -7 FF896 FPGA since the board available in our laboratory is based on this FPGA. The VLC Processor design presented in this paper utilizes 689,108 gates with 12 numbers of block RAMs. The maximum frequency of operation reported by ISE tool for this design is 124 MHz. The VLC processes a picture of size 640x480 pixels such as Apple at about 220 frames per second for a FIFO3 depth of 128 Kb. Extrapolating this result for higher frame size, the present FPGA implementation is capable of processing a picture of size 1600x1200 pixels at a frame rate of around 35 fps, thus meeting MPEG-2 standard.

5. Conclusions

A novel FPGA implementation of the core processor of a video encoder, namely, the Fast Variable Length Coder

has been presented. The simulation, synthesis, place and route results show that the designed Variable Length Coder is capable of processing very high resolution motion pictures of size 1600x1200 pixels at a frame rate of 30 per second meeting MPEG-2 standards. The reconstructed quality of the picture is quite good (33 dB) and, achieving a compression of 38. Presently, the integration of DCTQ and the VLC Processors targeted on a single FPGA is under progress. The proposed complete Video Encoder is being designed incorporating Rate Control in order to maintain a compressed, constant 100 Mbps serial throughput.

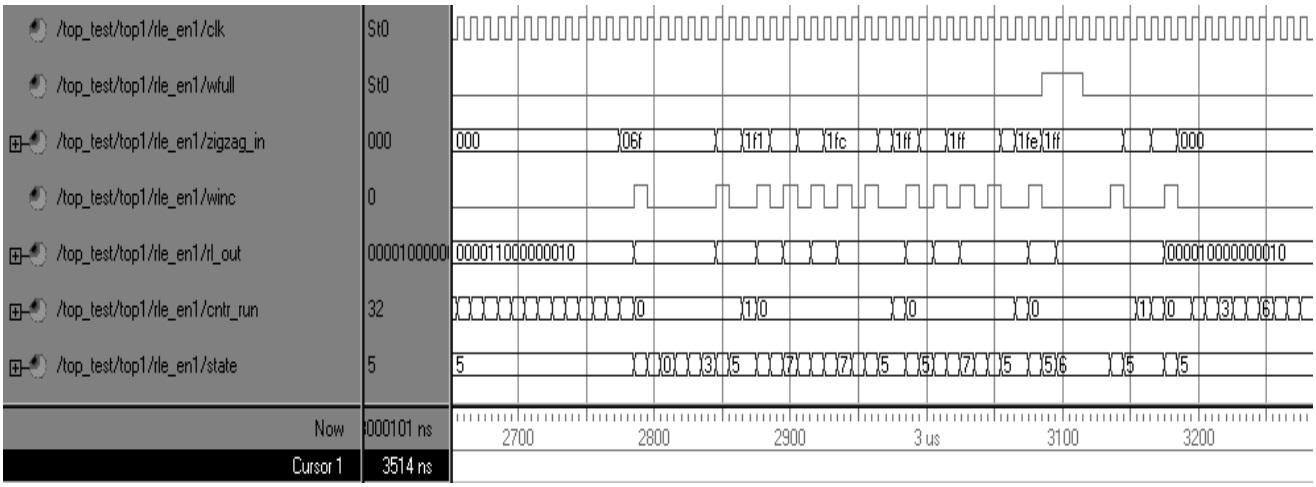


Figure 4 Snapshot of RLE Simulation

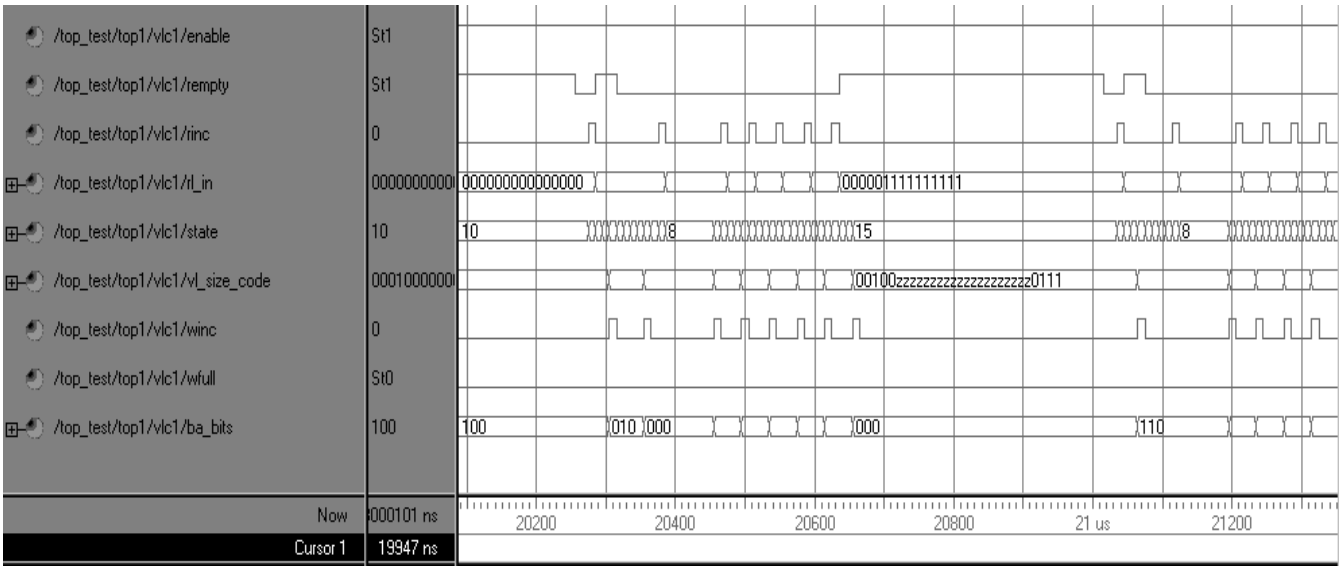
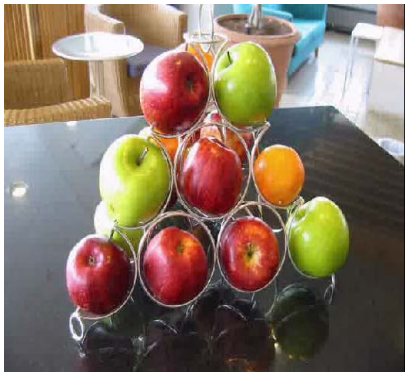


Figure 5 Snapshot of VLC Simulation

a Original Apple Frame



b Reconstructed Apple Frame

Figure 6 Sample reconstructed picture - Size: 640x480 Pixels, Compression effected: 38 and PSNR: 33 dB

References

[1] ISO/IEC MPEG-2 Standards for generic coding of moving pictures: Part 2, Video, 1998.

[2] N.I. Cho, S.U. Lee, "Fast Algorithm and Implementation of 2D-Discrete Cosine Transform", IEEE transactions on Circuits and Systems, Vol. CAS 38, pp. 297 - 305, Mar. 95.

[3] H. Park and V.K. Prasanna, "Area efficient VLSI architectures for Huffman coding", IEEE Transactions on circuits and systems, Vol. 40, No. 9, pp. 568-575, Sep. 1993.

[4] H.C. Chang, L.G. Chen, Y.C. Chang, S.C. Huang, "A VLSI Architecture Design of VLC Encoder for High Data Rate Video/Image Coding", IEEE International symposium on circuits and systems, Orlando, Florida, pp. iv398-401, May-June 1999.

[5] S. Ramachandran and S. Srinivasan, "A fast, FPGA-based MPEG-2 video encoder with a novel automatic quality control scheme," Elsevier, Journal of Microprocessors and Microsystems, UK, 25, pp. 449-457, 2002.

[6] S. Ramachandran and S. Srinivasan, "Design and Implementation of an EPLD-based Variable Length Coder for Real Time Image Compression Applications", IEEE International symposium on circuits and systems (ISCAS), Geneva, Switzerland, May, 2000.

[7] S. Ramachandran and S. Srinivasan,"FPGA Implementation of a Novel, Fast Motion Estimation Algorithm for Real-Time Video Compression", ACM/SIGDA Ninth International Symposium on Field Programmable Gate Arrays, Monterey, California USA, pp.213-219,2001.

[8] S. Ramachandran and S.Srinivasan, "A Novel Automatic Quality Control Scheme for Real-Time Image Transmission", VLSI Design Vol.14, pp.329-335, 2002.

[9] Jari Nikara, Stamatis Vasilliadis, Jarmo Takala, Petri Liuha, "Multiple-symbol parallel decoding for Variable Length Coders", IEEE Transactions on circuits and systems, Vol. 12, No. 7, pp. 676-685, July 2004.

[10] Jianjun Liu, Guofang Tu, Can Zhang and Yang Yang, "Joint source and channel decoding for variable length encoded turbo codes", EURASIP Journal on Advances in Signal Processing, Vol. 2008, Issue 1, 2008.

[11] Musy Stephane, "Variable Length Coder for Degraded Broadcast Channels", IEEE International Sympsium on Information theory, NICE, June 2007.

N.Venugopal obtained his ME degree from Bangalore University, India in 1998. He is pursuing his research work in Dr MGR University, Chennai. His research interest includes Video Processing, DSP applications, Communication, FPGA, Power electronics, Control systems and their applications. He is a member of IEEE and MISTE.

Dr. S. Ramachandran obtained his M. Tech. and Ph. D. from the Indian Institute of Technology, Kanpur and Madras respectively. He has wide academic as well as industrial experience of over 30 years, having worked as Professor in various engineering colleges as well as design engineer in industries. Prior to this, he has been with the Indian Institute of Technology Madras. His research interests include developing algorithms, architectures and implementations on FPGAs/ASICs for Video Processing, DSP applications, reconfigurable computing, open loop control systems, etc. He has a number of papers in International Journals and Conferences. He is the recipient of the Best Design Award at VLSI Design 2000, International Conference held at Calcutta, India and the Best Paper Award of the Session at WMSCI 2006, Orlando, Florida, USA. He has also written a book on Digital VLSI Systems Design, published by Springer Verlag, Netherlands (www.springer.com). His video lectures are accessible in You Tube.