IJCSC International Journal of Computer Science & Communication Vol. 1, No. 1, January-June 2010, pp. 57-65

Design of Architecture and FPGA Implementation of a Video Encoder

Venugopal N[1] & S. Ramachandran[2]

*Dr MGR University, Chennai and **National Academy of Excellence, Bangalore, India

E-mail: vgopal_m_e@yahoo.co.in, ramachandran_ns@yahoo.com

ABSTRACT

This paper proposes a novel VLSI architecture for Video Encoder, which processes high resolution video sequences at real time rates. The architecture has been realized using Verilog and implemented on an Xilinx XUPVP30 FPGA. The gate count of the implementation is approximately 800,000 including an output FIFO of size 128 K bits. It can process 1600x1200 pixels color motion pictures in 4:2:0 format at 30 frames per second as per MPEG-2 standard. The compression effected is typically 20 to 40 and the reconstructed picture is of good quality with a PSNR values of 32 dB or more. The main advantage of the architecture proposed is that it improves the throughput by over 30% compared to the earlier Encoder developed by one of the present authors. The proposed architecture of Video Encoder consists of Discrete Cosine Transform and Quantization Processor, Run Length Encoder, Variable Length Coder, Header Generator, Serializer, FIFOs and a Master Controller that co-ordinates all the activities of the encoder. The Video Encoder has also been coded in Matlab in order to validate the Verilog realization.

Keywords: Architecture, DCTQ, RLE, VLC, Video Encoder, Header, Verilog and FPGAs.

1. INTRODUCTION

MPEG-2 (Moving Pictures Experts Group standard) video codec systems are a core technology for the coming multimedia era. The MPEG-2 Video compression standard [1] is designed to produce broadcast quality video at higher bit rates of up to 100 Mbps. In the context of Video Compression, very few hardware implementations of MPEG-2 Encoder chain are available in the literature [2-4]. However, there are many efficient implementations of Discrete Cosine Transform and Quantization (DCTQ) processor [5-9] and Variable Length Coder (VLC) [10-19]. But none of them proposes a full Encoder chain with memory interfaces and controller implementation realized on a single chip such as an FPGA. For this reason, the proposed video encoder architecture has been implemented on a single FPGA. There are primarily two approaches in designing video encoder modules. One option is to use these modules as black boxes with a complex controller to handle the interfaces or glue logic. Another option is to use these techniques of efficient implementation with in-built interface controls to reduce the controller complexity. The latter option has been adopted in the present work.

The objective of the present work is to build an MPEG-2 Video Encoder for space applications in order to capture the separation stage of a launch vehicle for analyzing the movement of falling fragments. The Encoder should be capable of the following features:

- Process high resolution pictures of size: 1600 x 1200 pixels;
- Encode at high frame rate (30 fps);
- Process Color images in 4:2:0 format;
- Built-in Rate Control for constant bit stream output.

A high frame rate is desirable in order to capture as much details of the video as possible. Rate control is necessary so that the data can be transmitted over a serial channel at a constant bit-rate. Thus, the objective of the work presented in this paper is to design and build MPEG-2 Video Encoder which processes 1600x1200 pixels color motion picture at over 30 frames per second. However, Rate control is not envisaged in the present work.

Excluding this introductory section, the paper is organized as follows. The second section analyses three different architectures of video encoder in order to arrive at the optimum architecture for FPGA implementation. The third to fifth sections present the architectures of header generator, serializer and master controller respectively. This is followed by the results and discussions in the next section. The conclusions drawn are presented in the last section.

2. ANALYSIS AND SELECTION OF OPTIMUM ARCHITECTURE FOR VIDEO ENCODER

In the present section, three different architectures of video encoder are analyzed with the objective of selecting the right type of architecture that enables fast

implementation of the encoder. Fast implementation is indispensable if a high resolution picture such as that of size 1600x1200 pixels at high frame rate of 30 per second is required to be processed. A simple architecture of MPEG-2 encoder has been reported by one of the authors earlier [2]. The block diagram of the same implementation is presented in Fig. 1. As shown therein, a video sequence is input to the DCTQ module, where 64 numbers of quantized coefficients are generated for every picture block of 8x8 pixels. These coefficients are assigned variable length codes in the next module called RLE and VLC, thus bringing about compression. The header generator generates the picture information such as the picture size, frame rate, macro-block increment, etc. In order to maintain continuous processing of blocks in a frame of picture, the DCTQ coefficients were stored in double buffers marked Dual RAM 1. Thereafter, the header information and the VLC which is in parallel format are converted to a serial format in the next module called Serializer. The resulting serialized compressed bit stream is stored in a FIFO and subsequently transmitted over a serial channel.

The input to the DCTQ module is an 8 × 8 pixels block of a picture frame. Each pixel is represented in 8 bits for monochrome or 24 bits for Y, Cb (U) and Cr (V) components of a color picture. Its output is an 8x8 matrix (64 elements) representing the quantized DCT transformed coefficients. Each pixel is now represented in 9 bits (2's complement form). The run length encoding requires scanning of the DCTQ output in a zig-zag manner. Therefore, the RLE-VLC module can start processing the DCTQ output only after all the coefficients are available. Hence, the DCTQ output is stored in a dual RAM (Dual RAM 1 in Fig. 1). While DCTQ writes in one RAM, the RLE-VLC reads from the other RAM. After DCTQ output is ready, RLE-VLC starts scanning in a zig-zag manner. During scanning, RLE hits a non-zero AC coefficient after traversing none or a number of zeros. The VLC code for this non-zero level is computed. The code thus found is serialized. Thereafter, the RLE-VLC resumes its scanning. The sequence of operations for the Architecture 1 may be summarized as follows:

1. After DCTQ output is ready, RLE-VLC starts scanning the DCTQ coefficients in a zig-zag manner.
2. RLE hits a non-zero AC coefficient after traversing none or a few zeros.
3. The VLC code for the corresponding non-zero level is computed.
4. The code thus computed is serialized.
5. The RLE-VLC repeats the above sequence till all the coefficients of the block are processed.
6. The above process is repeated for all the blocks of a frame after frame.

The output of both RLE-VLC module and Header Generator module is a 29-bit word, in which the five most significant bits specify the number of bits to be transmitted from the remaining 24 bits. At any given instant of time, either the RLE-VLC or the Header Generator is enabled. The output of the Serializer is fed into a FIFO in order to generate serial output at a constant rate. The Master Controller maintains timing throughout the system.

In order to compute the VLC codes and execution time of Architecture 1, let us consider the quantized DCT coefficients of a picture block of size 8x8 pixels issued as the output of the DCTQ module and stored in Dual RAM 1. A sample matrix of DCTQ coefficients is shown in Fig. 2.

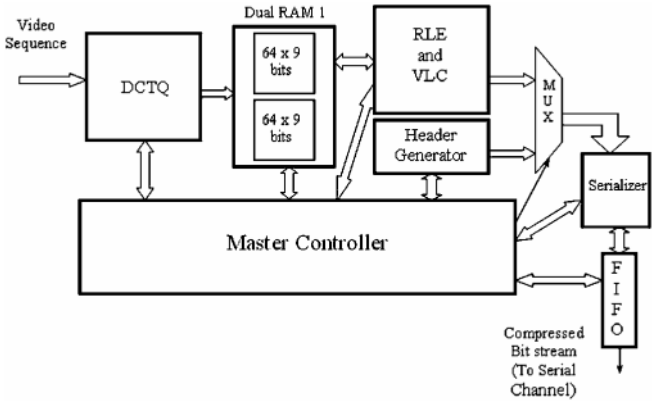

Fig 1: Simple Architecture 1 of Video Encoder

97	0	0	0	0	0	0	0
2	-3	0	0	0	0	0	0
4	-5	0	0	0	0	0	0
1	0	0	130	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	0	0	0	0	0	0	0

Fig 2: DCTQ Coefficients of a Sample Picture Block

The corresponding VLC code for the block of data is as follows:

DC differential = 97 – 128 = –31.

The value 128 is specified in the MPEG-2 standard.

VLC Code for the DC value of -31, assuming luminance is: 1110 0000 0 (a total of 9 bits). The rest of the DCTQ coefficients are AC coefficients and their VLC codes are tabulated in Table 1. These codes conform to MPEG-2 standard.

Table 1
VLC Codes of the AC Coefficients for the Sample Block

Run Length	*Level*	*VLC Code*	*VLC Code Size*
1	2	0001 100	7
0	4	0000 1100	8
0	–3	0010 11	6
3	–5	0000 0100 0011 1111 1111 1011	24
0	1	110	3
14	130	0000 0100 1110 0000 1000 0010	24
EOB		10	2

Table 2
Timing Analysis for Architecture 1

Run Length	*Level*	*VLC Code Size*	*No. of clock cycles required by RLE-VLC module**	*No. of clock cycles required to serialize***
1	2	7	2 + 1 + 1	7
0	4	8	1 + 1 + 1	8
0	-3	6	1 + 1 + 1	6
3	-5	24	4 + 1 + 1	24
0	1	3	1 + 1 + 1	3
14	130	24	15 + 1 + 1	24
EOB		2	39 + 1	2
			= 76	= 74

* The no. of clock cycles required = Scanning + VLC Encoding + Pass it to Serializer

** The no. of clock cycles required = No. of bits to be serialized

The number of clock cycles taken by each module processing adds up as there is no parallel processing in this scheme of integrated RLE-VLC. Timing Analysis for Architecture 1 is shown in Table 2. The execution time required for Architecture 1 is as follows:

No. of clock cycles required for computing DC differential = 3

No. of bits in DC differential VLC code = 9, requiring 9 clock cycles for execution in the Serializer.

Total no. of clock cycles required by RLE-VLC block = 76

Total no. of clock cycles required by Serializer = 74 + 9 (for DC code) = 83

Total no. of clock cycles required to process this block = 3 + 76 + 83 = 162.

Thus for a 256x256 pixels image, the number of clock cycles required may be calculated as follows:

Number of blocks = 256 x 256 / 64 = 1024 and

Number of clock cycles required = 1024 x 162 = 165888.

2.1. Performance Improvement of the MPEG-2 Encoder

It can be demonstrated that significant improvement can be obtained in terms of number of clock cycles required to process a frame by separating the RLE and VLC modules. The modified architecture for this scheme is shown in Fig. 3 and its timing analysis is shown in Table 3. The sequence of operations required for Architecture 2 is as follows:

1. After DCTQ output is ready, RLE starts scanning in a zig-zag manner from Dual RAM 1.
2. RLE hits a non-zero AC coefficient after traversing none or a few zeros and it writes the corresponding run-level in Dual RAM 2.
3. The VLC module reads run-level entry from Dual RAM 2, generates the corresponding VLC code and writes it into RAM 1.
4. Serializer reads from RAM 1 and serializes it.

Other steps are similar to that of Architecture 1 described earlier. Total execution time may be arrived as follows:

No. of clock cycles required for computing DC differential = 3

No. of bits in DC differential VLC code = 9

No. of clock cycles required by RLE module = 71

No. of clock cycles required by VLC module = 21

No. of clock cycles required by Serializer = 81 + 9 (for DC term) = 90

Total no. of clock cycles required for a block = 3+71 + 21 + 90 = 185.

In this architecture, while the run-level table of one block is being processed by VLC-Serializer, RLE would generate the run-level table of the next block.

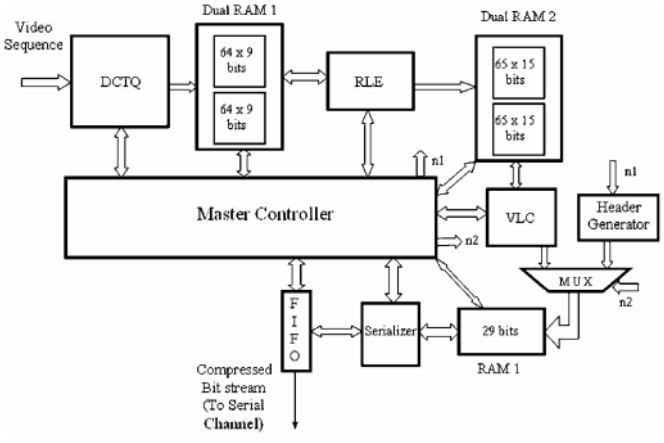

Fig 3: Modified Architecture

Table 3
Timing Analysis for Architecture 2

Run Length	*Level*	*VLC Code Size*	*No. of clock cycles required by RLE module* #	*No. of clock cycles required by VLC module*##	*No. of clock cycles required to serialize* ###
1	2	7	2 + 1	1+1+1	1+ 7
0	4	8	1 + 1	1+1+1	1+ 8
0	-3	6	1 + 1	1+1+1	1+ 6
3	-5	24	4+ 1	1+1+1	1 + 24
0	1	3	1 + 1	1+1+1	1 + 3
14	130	24	15 + 1	1+1+1	1 + 24
EOB		2	39+ 1 = 71	1+1+1 = 21	1 + 2 = 81

The no. of clock cycles required by RLE = No. of clock cycles required for Scanning + Writing into

Dual RAM 2

The no. of clock cycles required by VLC = No. of clock cycles required for Reading from Dual

RAM2 + Encoding + Writing into RAM 1

The no. of clock cycles required by Serializer = No. of clock cycles required for Reading from

RAM 1 and No. of bits to be serialized.

Thus after the first block is processed, only 3 + 21 + 90 = 114 clock cycles would be required by the VLC-Serializer to process each block.

Thus for a 256x256 picture, the number of clock cycles required would be as follows:

Number of blocks = 256 x 256 / 64 = 1024

Number of clock cycles required = 1 x 185 + 1023 x 114 = 116807.

The percentage saving in terms of the number of clock cycles required to process a frame in the new architecture is = ((165888 – 116807) / 165888) x 100 % = 30 % when compared to Architecture 1. Thus, with the separation of RLE and VLC module, by introducing the Dual RAM 2, a significant amount of time is saved in processing a frame. However, this architecture suffers from the following drawbacks:

1. Dual RAM 2 is memory-inefficient. The length of run-level table generated by RLE is of variable length. In one extreme case when there is no non-zero AC coefficient in DCTQ output, the length of the table will be 64. In another extreme case when all the AC coefficients are zero, the length of the table will be 1 (just the EOB). Therefore, the depth of a RAM in the Dual RAM 2 needs to be 65 including one extra entry to specify the length of the table. However, for most of the blocks the length of the table does not exceed more than 10. Thus, there is a huge amount of memory wastage.
2. The usage of dual RAM necessitates handshaking signals between the writer and the reader controls for that dual RAM. As we add more and more such modules to the Video Encoder, the handshaking becomes more and more involved, thus adversely affecting the functionality of the system.

2.2. An Alternative Architecture for the Video Encoder using FIFO Registers

One way of overcoming the drawbacks of our previous architecture is to replace the Dual RAM 2 with FIFO registers as shown in Fig. 4. Apart from the input and output ports, a FIFO register has been provided a 'full' and an 'empty' signal. When the 'full' signal is high, the writer stops writing into the FIFO. Similarly when the 'empty' signal is high, the reader stops reading from the FIFO. The FIFO eliminates the requirement of handshaking signals as follows:

Each major functional block in the architecture operates at its own frequency. For instance, the Serializer may slow down because of large number of bits it needs to serialize. It may be noted that the VLC code can vary between 2 and 24 bits. When FIFO 2 gets full, the VLC interprets the 'fifo_full' signal as a stop for processing the data. Accordingly, the VLC stops reading from FIFO 1, and in turn, when it gets full, the RLE also stops processing the data.

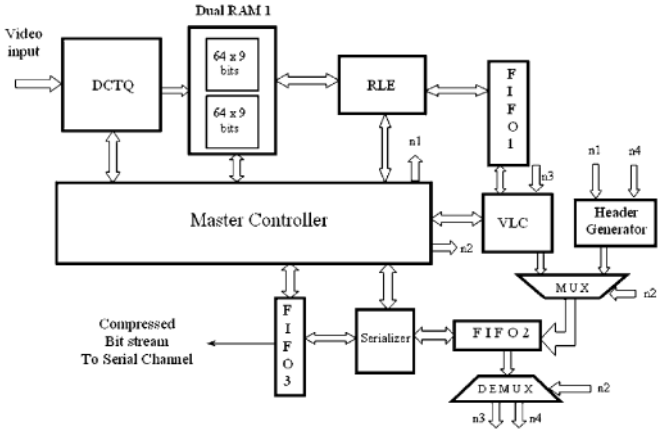

Fig 4 : Improved Video Encoder Architecture using FIFOs

This in turn forces the controller to instruct the DCTQ to hold the processing. Any speed bottleneck at any stage in the architecture propagates in the reverse order of the system owing to pipelining of various functional

modules and, the whole system works at the speed of the slowest module. As mentioned earlier, it is not possible to replace the Dual RAM 1 because of the requirement for zig-zag scanning. However, replacing the Dual RAM 2 between the RLE and the VLC with a FIFO mitigates the drawbacks of the architecture of Fig. 3 in the following ways:

1. There is a reduction of memory utilization from 1950 bits (2x65x15 bits) in Dual RAM 2 to only 120 bits (15x8) of FIFO. The size of the optimal FIFO register was arrived at by exhaustive experimentation. A FIFO larger than 120 bits shows no discernible advantage in speed over the 120 bits FIFO.
2. Maintaining proper timing between the write and the read control signals is a much less cumbersome process. No handshaking is required since the writing or reading of FIFO is made just by examining the full or empty status signals of the FIFO. Thus the complexity of the controller is significantly reduced.

The Video Encoder Architecture shown in Fig. 4 is, therefore, adopted for implementation in this work. The following sub sections describe in brief the architectures of DCTQ, RLE, VLC and FIFO developed earlier.

2.3. Architecture of DCTQ

A two-dimensional DCT is performed on small blocks (8 pixels by 8 pixels) of each component of the picture (Y, Cb and Cr) to produce blocks of DCT coefficients. The magnitude of each DCT coefficient indicates the contribution of a particular combination of horizontal and vertical spatial frequencies to the original picture block. The coefficient corresponding to zero horizontal and vertical frequency is called the DC coefficient. The remaining coefficients are called the AC coefficients. The DCT does not directly reduce the number of bits required to represent the block. In fact, for an 8x8 block of 8 bit pixels, the DCT produces an 8x8 block of 12 bit coefficients (the range of coefficient values is larger than the range of pixel values). The reduction in the number of bits follows from the observation that, for typical blocks from natural images, the distribution of coefficients is non-uniform due to spatial redundancy. The transform tends to concentrate the energy into the low-frequency coefficients and many of the other coefficients are near-zero. The bit rate reduction is achieved by not transmitting the near-zero coefficients and by quantizing and coding the remaining coefficients. One of the present authors has developed a fast algorithm for DCTQ and implemented the same on an FPGA [2, 8]. The same has been used in the present work for implementing the video encoder.

2.4. Architecture of the Run Length Encoder

The next functional block of the video encoder is the Run length encoder [19]. RLE reads from the double buffer (Dual RAM 1 in Fig. 4) in which is stored the DCTQ coefficients. RLE writes the processed output into the FIFO1 in Fig. 4. It writes the output in FIFO1 only if an empty location is available. Otherwise RLE automatically holds its operation without any controller signaling. The DCTQ coefficients block is scanned in a diagonal zig-zag manner starting from the DC coefficient to produce a list of non-zero quantized coefficient values, ordered according to the scan pattern. The number of zeros is counted before a non-zero AC coefficient is reached. The count of zeros is called the run length and the non-zero AC coefficient is called the level.

2.5. Variable Length Coder

The non-zero DCTQ values produced by RLE scanning are entropy coded next using a variable length code processor [19]. 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. VLC reads the DC differential and run-level from a FIFO in which the RLE module writes its output. Thereafter, VLC processes and writes the computed variable lengths for the DC differential as well as the AC coefficients into another FIFO. 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. The FIFO2 full/empty status is either sent to Header Generator or the VLC as the case may be. The Run length encoder and the VLC reported in Ref. [19] have been used for this work.

2.6. Architecture of First-in-First-out Memory

In a FIFO, data values are written sequentially into a FIFO buffer using one clock domain and the data values are sequentially read from the same FIFO buffer using another clock domain, where the two clock domains are asynchronous to each other. The buffer is a dual-port RAM addressed by Gray or binary counters. The FIFO full or empty status is generated by asynchronous comparison of the read and write pointers and are set asynchronously. In the video encoder implemented, FIFO1 is of depth 8 and width 15-bits (width of RLE output), FIFO2 is of depth 4 and width 29-bits (width of VLC output) and FIFO3 is of depth 128 K and width 1-bit (width of output of Serializer). 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 writer and the reader. Also 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. FIFO design of Ref. [20] has been utilized for this work.

3. ARCHITECTURE OF HEADER GENERATOR

At the beginning of each frame, the Header Generator 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 and the master controller issues the appropriate select signal. When a picture frame starts processing, the Header Generator outputs the header information.

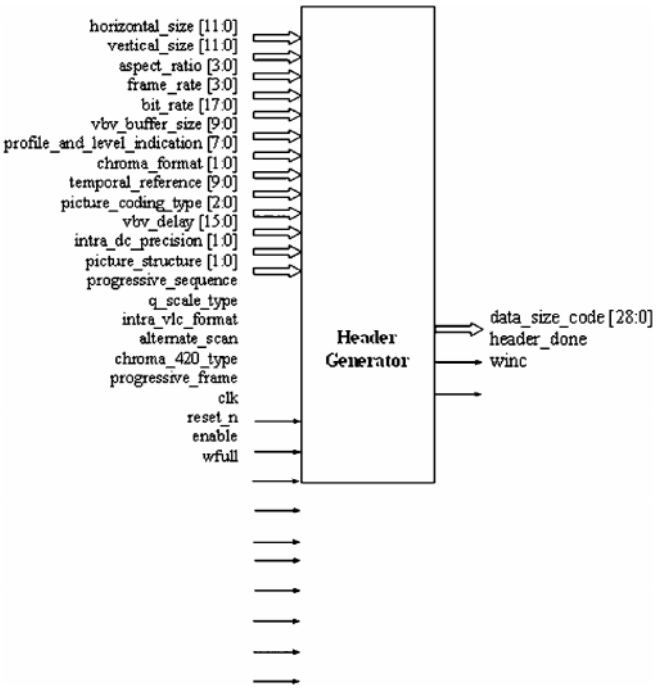

Fig 5: Architecture of Header Generator

The architecture of header generator is shown in Fig. 5. As shown therein, "horizontal_size [11:0]" and "vertical_size [11:0]", as the name implies, respectively specify the horizontal and the vertical sizes of the input frame. Similarly, "aspect_ratio [3:0]" specifies the aspect ratio of the input frame. Likewise, "frame_rate [3:0]" specifies the video sequence frame rate and "bit_rate [17:0]" specifies the output bit rate. "vbv_buffer_size [9:0]" specifies the Video Buffer Verifier size. The profile and level of the MPEG-2 stream is indicated by the signal "profile_and_level_indication [7:0]". The chroma format of the MPEG-2 stream is specified by "chroma_format [1:0]". Chroma format relates to macro-block structure. The temporal (pertaining to time) reference of the MPEG-2 stream is specified by "temporal_reference [9:0]".

The input "picture_coding_type [2:0]" specifies whether the frame is I, B or P type. "vbv_delay [15:0]" specifies the VBV Delay required to decode the video. The signal "intra_dc_precision [1:0]" specifies the intra DC precision required for resetting DC predictors. The "picture_structure [1:0]" specifies the picture structure of the frame. "progressive_sequence" specifies whether the sequence is progressive or not. "q_scale_type" specifies the Quantiser scale code required in inverse quantization at the decoder end. "intra_vlc_format" specifies the VLC table used in encoding the frames. "alternate_scan" specifies the zig-zag pattern used by RLE scanning the DCTQ coefficients."chroma_420_type" specifies the picture format. "progressive_frame" specifies whether the frame is progressive or not. The system clock is the input marked "clk". The header generator may be reset by using the asynchronous, active low signal "reset_n". Header operation can be held by de-asserting "enable" signal, if required. "wfull" signal is high when the FIFO in which header writes its output is full. "header_done" signal goes high after the header finishes outputting the header information. The header information is output into the "data_size_code [28:0]" pins. "winc" signal is write enable for the FIFO in which Header Generator writes its output.

4. ARCHITECTURE OF SERIALIZER

The architecture of Serializer is shown in Fig. 10. The 29 bits data issued out of VLC or header generator is fed into "data_size_code [28:0]" pins of the Serializer, where the parallel compressed video data is converted to a serial bit stream for onward transmission over a serial channel. Before transmission, the bit stream is buffered in FIFO 3 as described in Section 2 earlier. The bit stream may be regulated at a constant rate, say, 100 Mbps by incorporating rate control. This feature is yet to be designed. "rempty" input indicates that the FIFO from which Serializer reads is empty or not. Serializer operation can be held by de-asserting "enable" signal if required. Otherwise it functions normally. The Serializer, which is basically a shift register can be cleared if "reset_n" is asserted low. The "clk" input may be different from the system clock. The serialized output is available at "dout" pins and is valid when "dout_valid" is asserted. The output "rinc" is the write enable for the FIFO into which the Serializer writes its output. The salient features of the Serializer may be summarized as follows:

1. If the FIFO from which the Serializer reads its input (FIFO2) is not empty, Serializer will read one input at a time. Otherwise it keeps waiting for an input to arrive. The input "data_size_code" is of 29 bits width, where bits [28:24] specifies the number of bits that is required to be output at "data_size_code [23:0]" pins.

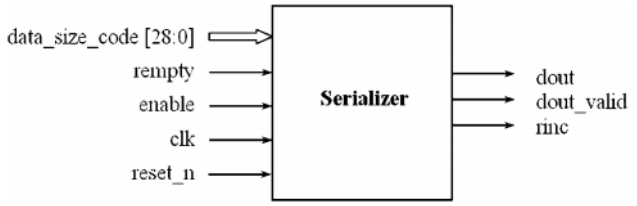

Fig 6: Architecture of Serializer

Example: For data_size_code = 00101 0000 0000 0000 0001 1001 1101, the Serializer output will be 11101.

2. The Serializer outputs one bit commencing from MSB at each positive edge of the Serializer "clk" till the required number of bits are serialized. The output is written into FIFO3. In case the FIFO3 is full, the controller holds the Serializer operation.

Steps 1 and 2 are repeated till the enable signal is high or the FIFO3 is not full. If "reset_n" is set, all the internal registers are cleared and the current Serializer processing is terminated.

5. ARCHITECTURE OF MASTER CONTROLLER

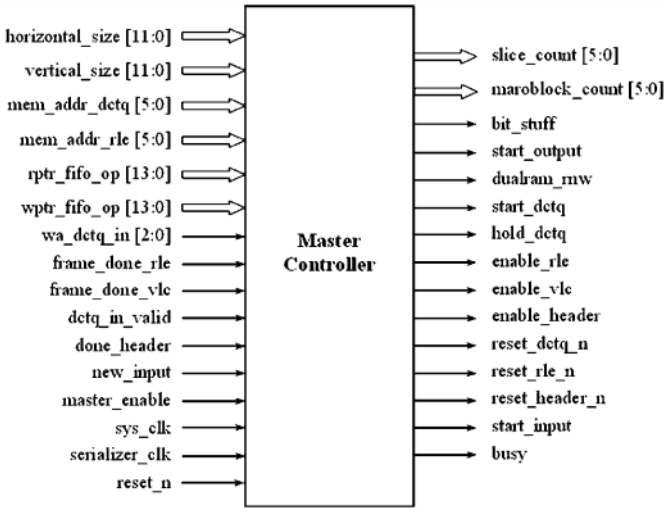

Fig 7: Architecture of Master Controller

The master controller controls the entire operation of the video encoder. It synchronizes the operation of DCTQ, RLE, VLC, Header Generator and the FIFOs. It takes the picture information and the various system clocks as input. It resets and enables various functional modules as required. The hold and bit stuff signals are also generated by the master controller when the FIFO levels breach the set levels. The master controller also switches the controls from the header generator to VLC and vice-versa at the appropriate instants. It also generates signals to switch between the two RAM's in dual RAMs. The architecture of Master Controller is depicted in Fig. 7. The input "horizontal_size [11:0]" is used to count the number of macro-blocks in a slice, while "vertical_size [11:0]" is used to count the number of slices in a frame. "mem_addr_dctq [5:0]" is the address of the memory location of the Dual RAM, where the DCTQ coefficient is to be written. Similarly, "mem_addr_rle [5:0]" is the address pointer to Dual RAM from which the RLE is to read its input. "rptr_fifo_op [13:0]" is the read address, whereas "wptr_fifo_op [13:0]" is the write pointer address of the output FIFO. These signals are used to implement rate control. "wa_dctq_in [2:0]" is write address for the block row input of DCTQ. "frame_done_rle" signals when the RLE finishes processing a frame. "frame_done_vlc" signals when the VLC finishes processing a frame.

The signal "dctq_in_valid" specifies when the input to the DCTQ is valid. "done_header" signals when the Header Generator finishes outputting header information. "new_input" signals when a new frame is ready at the input of the Encoder. "master_enable" is used for operation of the Encoder and can be held by de-asserting this signal if required. "sys_clk" is the system clock. "serializer_clk" is used as the clock input to the Serializer. "reset_n" is an asynchronous, active low signal to initiate system registers. "slice_count [5:0]" specifies the number of slices in a frame. "slice_count" equals vertical_size/16. "macroblock_count [5:0]" specifies the number of macro-blocks in a slice. "macroblock_count" equals horizontal_size/16. "bit_stuff" signal goes high when the fullness of the FIFO3 goes below a certain level. "start_output" is valid at each positive edge of output clock after it goes high. "dualram_rnw" signal selects one RAM for writing while enabling the reading of the other RAM in the Dual RAM. "start_dctq" is the start signal of the DCTQ module. The controller asserts "hold_dctq" when the DCTQ operation is required to be held. The signals "enable_rle", "enable_vlc" and "enable_header" are asserted for normal working of the RLE, VLC and Header Generator respectively. They are de-asserted when the corresponding operations are required to be held. "reset_dctq_n" is asserted when the DCTQ should be reset. Similarly, "reset_rle_n" and "reset_header_n" are asserted when the RLE and the Header Generator are required to be reset. "start_input" is issued to indicate the commencement of video input. The "busy" signal is asserted when the Video Encoder is busy processing a frame.

6. RESULTS AND DISCUSSIONS

The functional modules of the entire video encoder presented in earlier sections were simulated using Modelsim for a number of video sequences as well as still pictures such as tennis and apple. As an example, the waveforms for a sample picture, Apple of size 640x480 pixels is presented in Fig. 8. The compressed bit stream is issued from the Serializer module, whose architecture was presented in an earlier section. The

Serializer module is shown as "p2s1" and the serial bit stream output is depicted as "dout" in the figure. The serial data out "dout" is valid only when "dout_valid" is asserted high.

As can be seen from the two waveforms presented, the compressed bit stream commences at 46 ns starting with the header information: 0000 0000 0000 0000 0000 0001 and ends with data bitstream: 10 000000, the last bit "0" being transmitted at 5305626 ns. The difference between the start and the end of the bit stream, namely, 5305580 ns is the processing time for the Apple picture whose size is 640x480 pixels. This processing time is greater than the processing time of DCTQ although not presented here. Therefore, the slower module, namely, the VLC/Serializer determines the processing speed. This works out to 188 frames per second. The compressed bit stream generated by the Serializer module was written into an output file called "vlc.txt", which is superimposed on Fig. 8 b. It may be noted that the last eight bits of compressed data tallies with the bitstream 10000000 explained above. The "vlc.txt" file also reveals that the file contains 193656 bits. This means that the original Apple color picture that contained 640x480x24 bits (8 bits each for 3 color components Y, U and V) has been compressed to 193656 bits, meaning that the compression effected by the Verilog realization of the video encoder is 38.

Matlab codes were developed for the video codec in order to validate the Verilog design. The quality of the reconstructed picture is expressed as PSNR, for which also a Matlab code was developed. The PSNR value for Apple picture was 32 dB, reckoned as a good quality picture. Researchers world-wide accept that a reconstructed picture is of good quality if the PSNR value is 30 dB and above. The compressed bit stream (vlc.txt) obtained after simulation was converted to ".m2v" format file using another Matlab code and the same was run using third party MPEG-2 video player called Mplayer. A good quality motion picture was obtained, thus proving that the Verilog video encoder developed in this work conforms to MPEG-2 standard. Fig. 9 presents the original and the reconstructed pictures of "Apple". Elaborate experiments were conducted for reconstructing pictures of different sizes for various output FIFO depths using the implemented video encoder. The picture sizes used were primarily 352x288 pixels, 640x480 pixels and 1024x768 pixels. The FIFO depths experimented with was ranging from 16 Kilo bits to 128 Kilo bits. The picture frame rates achieved for these pictures are presented in Table 4.

Synthesis of the design was conducted using the Synplify tool. The ".edf" file generated by this tool was input into Xilinx Place & Route tool ISE 8.2i. These tool reports give the maximum frequency of operations possible as well as the gate count of the design. These results are tabulated in Table 5.

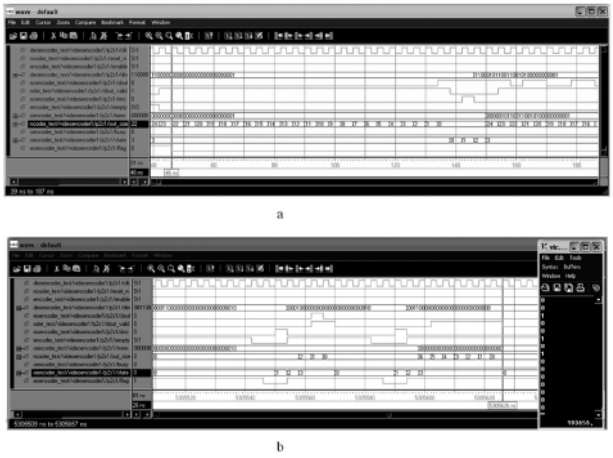

Fig 8: Verilog Simulation Results of a Sample Picture Apple using Modelsim, a. Waveform of Compressed Bit Stream at the Commencement of the Frame, b. Waveform of Compressed Bit Stream Towards the End of the Frame

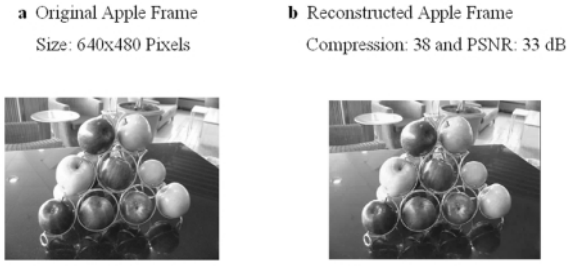

Fig 9: Verilog Simulation Results – Reconstructed Pictures a. Original Apple Frame, Size: 640x480 Pixels b. Reconstructed Apple Frame, Compression: 38 and PSNR: 33 dB

Table 4
The frame rate supported by the Implemented Video Encoder

Depth → Picture Size in pixels ↓	*FIFO Depth 16 Kb F/S*	*FIFO Depth 32 Kb F/S*	*FIFO Depth 64 KbF/S*	*FIFO Depth 128 KbF/S*
352 × 288	500	580	585	720
640 × 480	–	177	206	290
1024 × 768	–	–	89	144

Table 5
Synthesis/Place & Route Results for the Video Encoder Implemented on FPGA

Parameter → Module ↓	*Estimated Frequency system_clk (in MHz)*	*Estimated Frequency serialiser_clk (in MHz)*	*Estimated Frequency output_clk (in MHz)*	*Equivalent Gate Count*
DCTQ	141.5	-	-	118,540
RLE	204.0	-	-	4,216
VLC	172.7	-	-	6,437
Serializer	-	236.6	-	634
Header Generator	264.4	-	-	1,166
Encoder (FIFO depth 128 Kb)	102.2	195.6	223.1	807,648

7. CONCLUSIONS

A new architecture has been designed for the MPEG-2 Encoder and it is shown to exhibit good performance in terms of memory savings and speed. The Controller complexity is significantly reduced by the usage of FIFOs instead of Dual RAMs. The Encoder thus developed is compatible with the MPEG-2 standard. This was verified by decoding the output bit stream with a standard MPEG player. The simulation along with synthesis results shows that the desired objective of encoding high resolution images (1600x1200 pixels) at high frame rate (30 fps) has been achieved. The gate count of the designed video encoder was less than a million gates. The bit stream may be regulated at a constant bit rate by incorporating rate control. This feature is being designed currently.

REFERENCES

[1] ISO/IEC MPEG-2 Standards for Generic Coding of Moving Pictures: Part 2, Video, 1998.

[2] S. Ramachandran and S. Srinivasan, "A Fast, FPGA-based MPEG-2 Video Encoder with a Novel Automatic Quality . . . . . *Elsevier, Journal of Microprocessors and Microsystems*, UK, **25**, pp. 449-457, 2002.

[3] T. Akiyama, H. Aono, K. Aoki, K. W. Ler, B. Vdlismt, T. Araki, T. Morishige, H. Takeno, A. Sato, S. Nakatani, and T. Senoh, "MPEG Video Codec Using Image Compression DSP", *IEEE Transactions on Consumer Electronics*, **40**, No. 3, AUGUST 1994.

[4] Geng-Lin Chen, Jyh-Shin Pan, and Jia-Lung Wang "Video Encoder Architecture For MPEG 2 Real Time Encoding", *IEEE Transactions on Consumer Electronics*, **42**, No. 3, AUGUST 1996.

[5] 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. 91.

[6] C. L. Wang and C. Y. Chen, "High Throughput VLSI Architectures for the 1-D and 2-D Discrete Cosine Transforms", *IEEE Trans. Circuits Syst. Video Technol.*, **5**, pp. 31-40, 1995.

[7] Yung-Pin Lee, Thou-Ho Chen, Liang-Gee Chen, Mei-Juan Chen and Chung-Wei Ku, "A Cost-effective Architecture for 8x8 2-D DCT/IDCT using Direct Method", *IEEE Trans. Circuits Syst. Video Technol.*, **7**, 1997.

[8] S. Ramachandran, S. Srinivasan and R. Chen, "EPLD-based Architecture of Real Time 2D-Discrete Cosine Transform and Quantization for Image Compression", *IEEE International Symposium on Circuits and Systems (ISCAS '99)*, Orlando, Florida, May-June 1999.

[9] Tian-Sheuan Chang, Chin-Sheng Kung and Chein-Wei Jen, "A Simple Processor Core Design for DCT/IDCT", *IEEE Trans. Circuits Syst. Video Technol.*, **10**, pp. 439-447, 2000.

[10] H. Park and V.K. Prasanna, "Area Efficient VLSI Architectures for Huffman Coding", *IEEE Transactions on Circuits and Systems*, **40**, No. 9, pp. 568-575, Sep. 1993.

[11] 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.

[12] 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.

[13] Jari Nikara, Stamatis Vasilliadis, Jarmo Takala, Petri Liuha, "Multiple-symbol Parallel Decoding for Variable Length Coders", *IEEE Transactions on Circuits and Systems*, **12**, No. 7, pp. 676-685, July 2004.

[14] 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.

[15] Musy Stephane, "Variable Length Coder for Degraded Broadcast Channels", *IEEE International Symposium on Information Theory*, NICE, June 2007.

[16] Jin-young Yang, Jinwoong Kim, Sang Gyu Park "A Variable Length Coding ASIC Chip for HDTV Video Encoders", *IEEE*, 1997.

[17] Aslan Tchamkerten and I. Emre Telatar, "Variable Length Coding over an Unknown Channel" *IEEE Transactions on Information Theory*, **52**, No. 5, May 2006.

[18] Qiang Wang, Debin Zhao and Wen Gao, "Context-Based Adaptive Variable Length Coding for Video DCT Blocks Part II - Practical Scheme", Express Letters, ICIC International, **3**, Number 2, June 2009.

[19] N. Venugopal and S. Ramachandran, "Design of FPGA Implementation of Fast Variable Length Coder for a Video Encoder," *International Journal of Computer Science and Network Security*, **9**, No. 7, pp. 178-184, July 2009.

[20] Clifford E. Cummings Peter, "Simulation and Synthesis Techniques for Asynchronous FIFO Design with Asynchronous Pointer Comparisons", Xilinx, Inc.