An FPGA-Based Video Compressor for H.263 Compatible Bit Streams

G.Lienhart, R. Männer
Dept. for Computer Science V
University of Mannheim,
B6-26, D-68131 Mannheim, Germany
+49-621-181-2642
lienhart | maenner@ti.uni-mannheim.de

R. Lay, K.H.Noffz
Silicon Software GmbH
M2-16, D-68161 Mannheim,
Germany
+49-621-181-2666
lay | noffz@silicon-software.com

ABSTRACT
This paper presents an FPGA architecture for video encoding according to the H.263 standard for video teleconferencing systems. The implementation is based on an off-the-shelf FPGA[1] and is embedded in a PCI plug-in card[2] with on-board SRAM plus external SRAM. The most complex part of the H.263 protocol, a base-line encoder, was implemented. The strategies, which have been applied to build the complex encoding operations, are treated in this paper. The complete application is able to operate at 30 MHz. This leads to a maximum compression speed of 120 Mbit/s allowing simultaneous real-time operation of several video streams in a single reconfigurable chip. Enhanced coding options can also become implemented with present-day FPGAs. The use of FPGA technology enables the adaptation of hardware to various protocols and environments by software and therefore saves development time and hardware costs.

Categories and Subject Descriptors
B.6.1 [**Hardware - Logic-Design - Design Styles**]

General Terms
Performance, Design.

Keywords
Video Compressor, FPGA, H.263, Distributed Arithmetic.

1. INTRODUCTION
Low bit rate coding is essential for video conference and video telephony systems. The ITU proposed the H.261 and H.263 standard for these affairs. In order to achieve high bit rate reduction under the constraint of the highest possible picture quality these source-coding schemes are very sophisticated and require hardware processors of high complexity. Thus, application specific integrated circuits for video coding are normally used for high-performance compression systems [1].

Recently, the freely configurable FPGA technology became capable of executing complex video compression algorithms with high performance. In this paper an FPGA implementation for H.263 bit streams is presented. Special interest is taken in the implementation of the forward and inverse discrete cosine transformation of the compression scheme. The solution with distributed arithmetic is presented. The efficient realization of the quantization and dequantization unit in a serial manner is shown in principle. The concept of the entropy coder is also illustrated.

2. VIDEO COMPRESSION SCHEME
Bitrate reduction of the video data is possible because of redundant and irrelevant information in the video signal. Basically, there are two sources of redundancy. Within a video frame there is a spatial redundancy because of the statistical correlation between contiguous pixels. Between preceding frames, there is temporal redundancy due to slow changing scenes in a video stream. Irrelevance of information in the video signal means that such information is insignificant for human visual perception.

Reduction of temporal redundancy is performed through predictive coding, and statistical decorrelation is done by transforming the blocks of a partitioned image by the use of the discrete cosine transformation (DCT). Irrelevance reduction is achieved by quantizing the DCT results adapted to visual properties of the human eye.

For predictive coding, successive frames are combined by building differential interframes. H.263 enables different motion compensation strategies to improve the prediction. Here, no motion compensation is applied, so that predictive coding simply means building the difference between a current pixel of the present frame and the corresponding pixel of the previous frame.

For the decorrelating transformation H.263 defines the application of the DCT on 8x8 pixel blocks of a frame, as in many other image compression schemes, too. Quantization of the transformation results (Q) is achieved by integer division by user defined parameters. These parameters are chosen in such a manner that information reduction can become achieved with a minimum of visual artifacts. The quantized values become arranged in order of ascending frequency (so called zigzag-rearranging) and then entropy coded. Therefore, run length coding is used, where long chains of "0's" that occur in this coefficient stream are efficiently represented. A Huffman coding scheme finally maps these run length symbols to variable length codes.

[1] An FPGA of the widespread XILINX XC4000X Series has been used (XC4085XLA).

[2] The "microEnable" FPGA Coprocessor board of Silicon-Software has been used [10].

FPGA'01, February 11-13, 2001, Monterey, California, USA.

The entropy coding is reversible, however the quantization and in practice also the DCT are irreversible. In order to have the same prediction at both the receiver and the transmitter, a reconstruction path containing dequantization (Q^{-1}) and inverse DCT (IDCT) shall always be incorporated into the encoder. Therewith the prediction of a current frame is based on the reconstructed image of the previous frame.

A block diagram of the video encoder is shown in Figure 1.

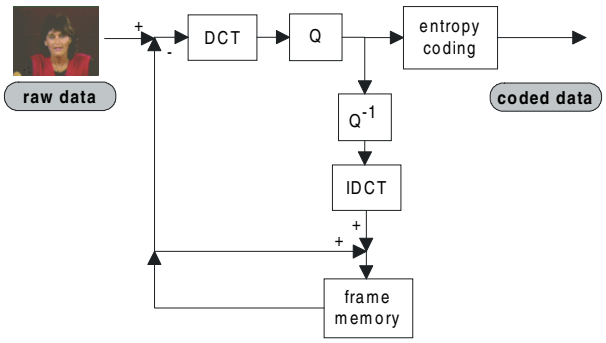

Figure 1. Baseline video compression scheme.

3. THE ARCHITECTURE

3.1 Forward and inverse DCT

In this section, we will have a close look on the arithmetic used for calculating the forward and the inverse DCT. For these transformations on 8x8 pixel blocks (2D-DCT and 2D-IDCT) distributed arithmetic[3] is used (see [1]-[3]). This leads to a bit-serial computation where only 16 word look-up tables (ROMs) and accumulators but no multipliers need to be utilized. This allows an implementation, which is very resource efficient on FPGAs as their architecture is well suited for accumulators and small ROMs. Especially ROMs with 16 words accessed through a four bit address are optimal because 16x1 ROMs are the basic building blocks for logic in commonly used FPGAs. Below, we give a brief derivation of the usage of distributed arithmetic for calculating the 2D-DCT.

For H.263 the 2D-DCT is defined as:

$$X \xrightarrow{8x8\ \ 2D-DCT} Y$$

$$y_{kl} = \frac{c(k)c(l)}{4} \sum_{i=0}^{7} \sum_{j=0}^{7} x_{ij} \cos\left(\frac{(2i+1)k\pi}{16}\right) \cos\left(\frac{(2j+1)l\pi}{16}\right)$$

$$with \quad c(i) = \begin{cases} \frac{1}{\sqrt{2}} & for\ i = 0 \\ 1 & else \end{cases} \qquad (1)$$

The 2D-DCT can be computed by the use of one-dimensional DCTs (1D-DCTs) of the rows and subsequent 1D-DCTs of the columns. This is possible because of the separability of the 2D-DCT. The 1D-DCTs can be expressed as follows:

$$y_l = \frac{c(l)}{2} \sum_{m=0}^{7} x_m \cos\left(\frac{(2m+1)l\pi}{16}\right) = \sum_{m=0}^{7} a_m{}^l x_m$$

$$with\ a_m{}^l = \frac{c(l)}{2} \cos\left(\frac{(2m+1)l\pi}{16}\right) \qquad (2)$$

Utilizing the property of the factors a_m^l to be symmetric in m, we only need to sum up four product terms:

$$y_l = \begin{cases} \sum_{m=0}^{3} a_m{}^l (x_m + x_{7-m})\ for\ l\ even \\ \sum_{m=0}^{3} a_m{}^l (x_m - x_{7-m})\ for\ l\ odd \end{cases} \qquad (3)$$

For easier writing, we define u_m as shortcuts for the sums and differences of x_m:

$$y_l = \sum_{m=0}^{3} a_m{}^l u_m$$

$$with\ u_m = \begin{cases} (x_m + x_{7-m})\ \ for\ l\ even \\ (x_m - x_{7-m})\ \ for\ l\ odd \end{cases} \qquad (4)$$

We can write the u_m as a sum of weighted bits (note: B is the data width of u_m):

$$u_m = -u_m{}^{(0)} + \sum_{j=1}^{B-1} u_m{}^{(j)} 2^{-j} \qquad (5)$$

The change of summing order in m and j as expressed in (6) characterizes the distributed arithmetic scheme in which the initial multiplications are distributed to another computation pattern.

$$y_l = \sum_{m=0}^{3} a_m{}^l \left[-u_m{}^{(0)} + \sum_{j=1}^{B-1} u_m{}^{(j)} 2^{-j} \right]$$

$$= \sum_{j=1}^{B-1} \left[\sum_{m=0}^{3} a_m{}^l u_m{}^{(j)} \right] 2^{-j} - \sum_{m=0}^{3} a_m{}^l u_m{}^{(0)}$$

$$y_l = \sum_{j=1}^{B-1} F(a^l, u^{(j)}) 2^{-j} - F(a^l, u^{(0)}) \qquad (6)$$

$$with\ F(a^l, u^{(j)}) := \sum_{m=0}^{3} a_m{}^l u_m{}^{(j)}$$

For any index l the $F(a^l,u^{(j)})$ can become precalculated and stored in a ROM with 16 entries. So bit-serial evaluation of the summation formula (6) for y_l only requires one "rom-and-accumulator" (RAC) element for each l as shown in Figure 2, with preprocessed u_m (sums or differences of x_m and x_{7-m}, m=0..4) as bit-serial input. One 1D-DCT thus consists of eight RACs plus a bit-serial preprocessor which calculates the $u_m^{(j)}$. In Figure 3 you can see the serial preprocessor unit connected with the RAC devices. After B steps (note that B is the data width of u_m), the results of the 1D-DCT appear parallel at y_0 to y_7.

Before applying the second 1D-DCT in the same manner, the 8x8 matrix of the results of the first 1D-DCT must become transposed. This is done by a bit-serial transposition network using the recursive transposition technique as described in [3]. The principle is to apply

[3] Distributed arithmetic is a key technology for numerous digital signal processing applications (see e.g. [7]).

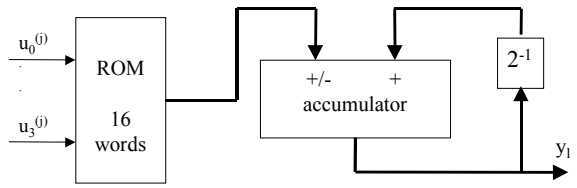

Figure 2. Rom-and-accumulator element (RAC).

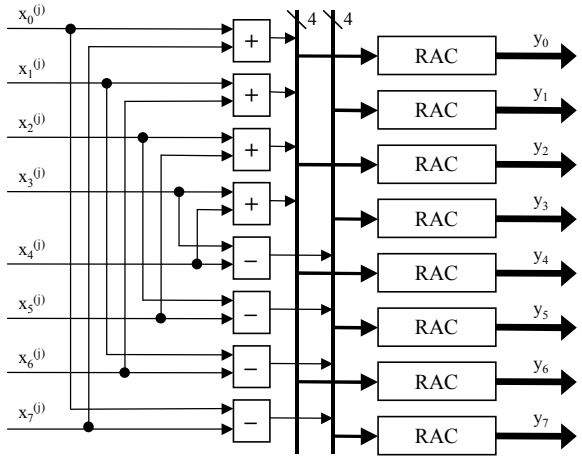

Figure 3. Scheme of the 1D-DCT circuit.

successive 2x2 transpositions on 1x1, 2x2 and finally 4x4 word sub-blocks, assuming bit-serial input of the 8x8 blocks of data. These serial 2x2 transpositions are done by delaying some data words relative to others and switching the data path inside an elementary transposition cell periodically. Figure 4 shows the interconnection scheme of the transposition circuit, which is composed of shift registers and crossing switches. Here a word length B of 16 for the data words was assumed but any other word length is possible. C_0, C_1 and C_2 are determined to toggle every B, 2*B and 4*B clock periods.

By utilization of the elementary on-chip RAM cells of the FPGA for serial shift registers, a very efficient design resulted. For serial input to the transposition unit, eight additional parallel-to-serial converters are employed. Figure 5 shows the flowchart of the complete 2D-DCT unit assembled of the discussed modules.

To show that the same processing method is applicable for implementing the inverse 2D-DCT (2D-IDCT), which is separable just as in the case of the 2D-DCT, the formulas for calculating the one dimensional inverse DCT are given in (7).

$$
\begin{aligned}
x_l &= \sum_{j=0}^{7} y_j \frac{c(j)}{2} \cos\left(\frac{(2l+1)j\pi}{16}\right) \\
\frac{1}{2}\mathrm{u}_l &= \frac{1}{2}(x_l + x_{7-l}) \\
&= \sum_{m=0}^{3} y_{2m} \frac{c(2m)}{2} \cos\left(\frac{(2l+1)2m\pi}{16}\right) \\
&= \sum_{m=0}^{3} a_l^{\,2m} y_{2m} \\
\frac{1}{2}\mathrm{v}_l &= \frac{1}{2}(x_l - x_{7-l}) \\
&= \sum_{m=0}^{3} y_{2m+1} \frac{c(2m+1)}{2} \cos\left(\frac{(2l+1)(2m+1)\pi}{16}\right) \\
&= \sum_{k=0}^{3} a_l^{\,2m+1} y_{2m+1}
\end{aligned}
\qquad (7)
$$

$$
\left.
\begin{aligned}
x_l &= \frac{\mathrm{u}_l}{2} + \frac{\mathrm{v}_l}{2} \\
x_{7-l} &= \frac{\mathrm{u}_l}{2} - \frac{\mathrm{v}_l}{2}
\end{aligned}
\right\} \quad l = 0..3
$$

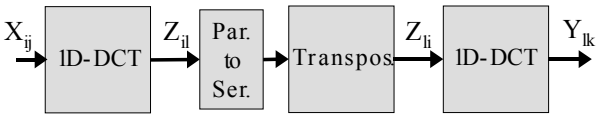

Figure 5. Composition of the 2D-DCT with usage of 1D-DCT operators.

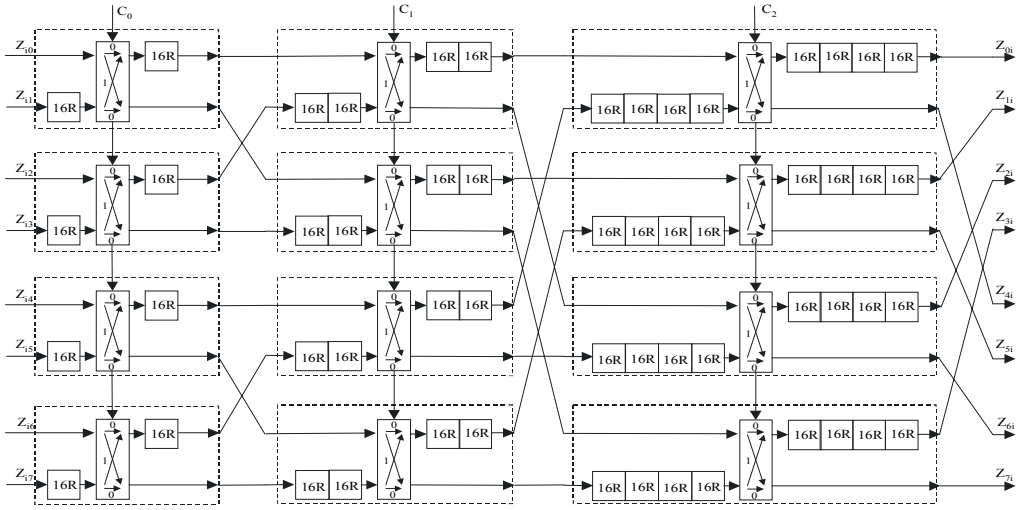

Figure 4. Bit serial transposition operator.

The values (½ u_l) and (½ v_l) are calculated just as shown in (6), wherefore the same RAC-units as in Figure 2 can be utilized. Instead of a preprocessing unit in the case of the 1D-DCT, here a postprocessing device is needed, which calculates the results x_l out of the u_l and v_l.

According to resource limitations of the used FPGA, the following word lengths were applied for the 2D-DCT and 2D-IDCT in our implementation: For the RAC elements of Figure 2 the ROM coefficients were stored with 12 bit accuracy and the accumulators operated with 19 bits. The transposition unit transported numbers with 16 bit precision. With these word lengths the accuracy of the resulting 2D-IDCT could not meet the IEEE specification [8]. To meet the specification with this IDCT architecture, ROM coefficients of 20 bit accuracy, 28 bit accumulators and 22 bit numbers in the transposition unit are needed. This would approximately increase the resource utilization of the IDCT by 50 %. Despite of having less accuracy than specified, the resulting image quality of the whole compression system was quite satisfactory.

3.2 Quantization and Dequantization

As mentioned in section 2 the quantizer principally performs integer divisions. Correspondingly the inverse quantizer calculates integer multiplications. Both operators work with serial arithmetic. This leads to a very simple data path between DCT and IDCT without expendable data distribution and buffering, particularly because the IDCT needs bit-serial input. Furthermore the applied serial dividers and multipliers are both resource efficient and fast. Admittedly synthesizing the case differentiation of the formula for the inverse quantization was more complex compared to an implementation with bit-parallel operators. Moreover reversing the order of bits both in the quantizer and inverse quantizer became necessary. But still the advantages of serial computation preponderated because the elements, which were additional to the dividers and multipliers, could have been mapped very efficiently onto the FPGA resources. Figure 6 shows the principal flowchart of the quantizer and its inverse.

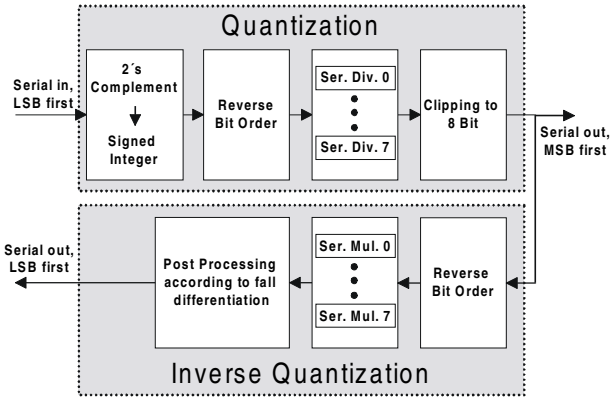

Figure 6. Scheme of the quantization and inverse quantization unit.

3.3 Entropy Coding

The entropy-coding unit (Figure 7) begins with a module, which arranges the quantized coefficients in order of ascending frequency (zigzag rearranging). This is done by the use of on-chip RAM (dual port RAM mode).

The rearranged data is the input for a variable length coder (VL-Coder). The H.263 recommendation specifies run length coding. Accordingly run length codes consist of a run number, which indicates the count of zeros preceding a non-zero coefficient, the value of the non-zero coefficient itself (level) and an information bit, whether this coefficient is the last of the current block. The VL-Coder maps these three-dimensional run length to variable length codes (VL-codes). H.263 defines a code table with 101 VL-codes. These codes are stored in an on-chip look-up table (LUT). For mapping, a resource optimized code-assign module was constructed. In order to save on-chip memory and logic resources for multiplexing, the LUT became partitioned in seven sub-LUTs. Thus it was possible to design an easy and cheap LUT-addressing, where the memory for the LUTs were utilized with a saturation of more than 70% (a straightforward implementation of the LUT would result in only 10% saturation).

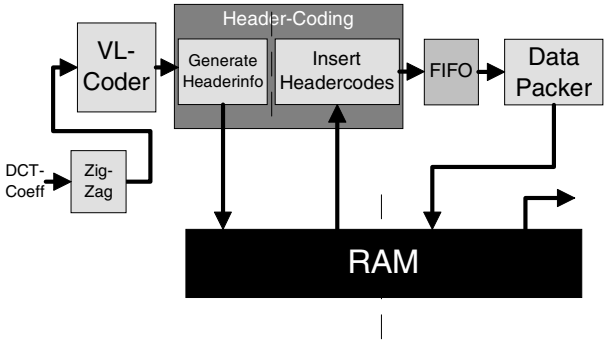

Figure 7. Scheme of the entropy-coding unit.

In H.263, picture data is divided in subunits containing the codes of 6 succeeding 8x8 pixel blocks (called macro block). Preceding the VL-codes for these picture units, a header has to be inserted, consisting of coding information related to the macro block. But the generation of these headers depends on coefficient data of the whole macro block, which is still not available at the time of emission of the first VL-code of a macro block. So an individual RAM was utilized for temporary storing of variable length codes. The modules GenerateHeaderinfo and InsertHeadercodes of Figure 7 generate the correct header information related to the coefficient data, and produce the codes for the macro block headers.

Packing the header- and VL-codes to constant word-length completes the processing. Here the codes become packed to 32 bit words corresponding to the data width of the RAM and the PCI interface. After packing, the results are stored in the same RAM which was utilized for buffering the VL-codes before header insertion. To balance the write access to the RAM, a FIFO-module was inserted between the header coding unit and the data packer.

3.4 Total System

As mentioned above the used hardware platform consists basically of a PCI plug-in card with one FPGA chip, on-board SRAM and additional external SRAM located on an add-on card.

Figure 8 shows how the encoding modules inside the FPGA are connected and embedded in the FPGA periphery.

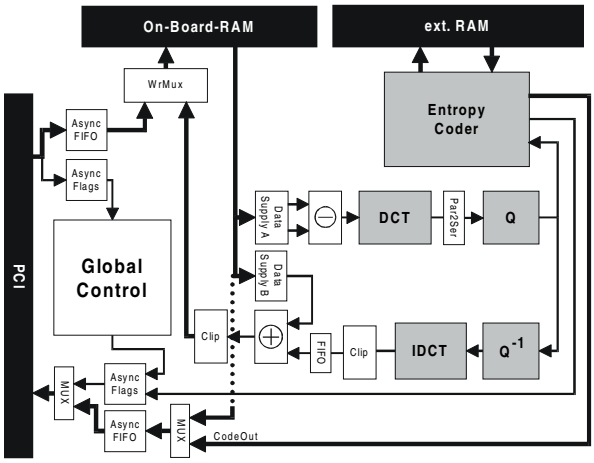

Figure 8. Simplified scheme of the encoder circuit

Interchange of data between FPGA and the PCI interface is controlled by asynchronous FIFOs and Flags according to the different clock systems of the FPGA and the PCI-interface. Before processing, picture data is stored in on-board RAM. This RAM is also used for storing the results of the reconstruction path of the encoder. The gray signified modules of the encoding kernel are in accordance with the blocks of the scheme in Figure 1. Preceding the DCT, a subtraction module enables calculating the differences of the current picture and the reconstructed frame of the previous picture to encode interframes. Following the IDCT, an adder unit supports generation of new reconstructed images out of the previous reconstructed frame and the current reconstructed interframe. Two clipping stages are necessary to avoid overflow errors.

The entropy coder utilizes an individual RAM (external RAM) as mentioned in the preceding section.

4. PERFORMANCE

The whole design fitted in an off-the-shelf FPGA[4]. With the used FPGA, the design is able to operate at 30 MHz clock frequency (at some points in the design, doubled frequency of 60 MHz has been used, e.g. for realizing dual port memory access). As the DCT is able to process one 8x8 pixel block of 8 bit values in 128 clock cycles, a maximum compression speed of 120 Mbit/s results. This peak performance translates to a compression rate of 98.6 CIF[5] frames per second. Hence three image sequences at 30 frames per second can become compressed simultaneously.

The compression results were compared with the output from a software reference encoder[6], whose compression options were disabled. Except for a slightly different generation scheme of interframes, the functionality of the software encoder was comparable to our application[7]. As input, the popularly used "Miss America" standard image sequence was taken.

[4] 89 % silicon area of the used Xilinx FPGA XC4085 XLA were utilized.

[5] The CIF standard defines images with 352x288 pixels stored in 4:1:1 Y-Cb-Cr color format.

[6] The Telenor TMN5 encoder has been used.

[7] But the speed of the reference encoder was not representative for general software encoders because it was not speed-optimized. Particularly it could not perform real-time compression.

Several measurements of signal-to-noise ratios for different quantization values are listed in Table 1.

Table 1. Mean SNR of the first 30 pictures from the "Miss America" standard sequence with use of inter-coding.

Quant	4	8	12	16	20
FPGA-Encoder					
SNR-Y/dB	40.6	38.1	36.6	35.4	34.5
SNR-Cb/dB	40.5	38.6	37.5	36.8	35.2
SNR-Cr/dB	42.6	39.6	37.8	36.2	35.2
Software-Encoder					
SNR-Y/dB	40.8	38.5	37.2	36.3	35.5
SNR-Cb/dB	40.6	38.8	37.8	37.1	36.6
SNR-Cr/dB	42.9	40.1	38.4	37.1	36.0

The quantities in Table 1 evidence the subjective visual impression that the image quality of the decompressed bit stream of the FPGA based encoder is nearly as good as it is with the output of the software encoder.

Table 2 faces the achieved compression ratios of the FPGA encoder with the results of software encoding for different quantization values. The ratios of the hardware video compressor are noticeable below those of the software, especially for high "Quant"-Values. This is basically a result of the better conjunction scheme between successive frames applied by the software. The software utilizes a prediction scheme which effectively has the functionality of smoothing a reconstructed frame before combining it with the current frame to an interframe. This feature has not been implemented in the FPGA based encoder due to the resource restrictions of the target FPGA.

Table 2. Compression ratios for the complete "miss America" sequence with 111 pictures.

Quant	4	8	12	16	20
Uncompressed	16484 KB				
Compressed with FPGA	469 KB	136 KB	88 KB	69 KB	59 KB
Compression ratio	35:1	121:1	187:1	239:1	279:1
Compressed with Software	288 KB	75 KB	45 KB	34 KB	28 KB
Compression ratio	57:1	220:1	366:1	485:1	589:1

5. CONCLUSIONS

We described an architecture well-suited for building the modules of a video compression system with FPGA technology. The application of distributed and bit-serial arithmetic has shown to be optimal for our FPGA implementation. A design resulted, where a complete baseline encoder for H.263 compatible bit streams was successfully implemented in a single off-the-shelf FPGA.

The single chip system has a compression speed like high-performance hardware systems. Tests of the encoder evidenced an image quality of the decoded bit streams, which was similar to

software compression results, but the compression ratio was not as high as that of software encoding.

With enhanced FPGAs the DCT and IDCT may become implemented with sufficient accuracy to get an image quality as good as with software encoding. By implementing an improved prediction scheme, it will also be possible to close the gap of compression ratio between an FPGA application and software. Present-day FPGAs already have enough logic resources for realizing this features.

In the described application the image data was transported through the PCI-Interface, so the FPGA operated as a coprocessor. But it is also possible to attach an image acquisition unit directly with the FPGA board (e.g. any camera interface). The interface to such a unit can become implemented inside the FPGA and exchanged dynamically depending on the requirements of the system. In this way the video compression hardware may work stand-alone, disengaging the CPU for other tasks. Moreover data channel bottlenecks between high speed camera and host computer may become resolved with such a system.

Above all, the use of an FPGA for performing the video compression algorithm enables the individual adaption of the core algorithm with changing ancillary conditions. For example different designs optimized for speed, power consumption, accuracy or different image characteristics can become applied dynamically.

The new possibility of using freely configurable FPGAs for video compression applications opens the door to novel flexible applications. For example direct connection of various cameras to fixed compression hardware becomes possible. Also platform independent compression and transcoding systems with exchangeable interfaces, loaded by software or ROMs, may become designed with the use of FPGAs.

Using Field Programmable Gate Arrays enables a variety of adaptive and intelligent high speed video compression systems as the behavior of the complete circuit is configurable and selectable by software. At last the process of fixing bugs in hardware gets reduced to modify software, which means reduction of development time and minimization of the risk of hardware failures.

6. REFERENCES

[1] P. Pirsch, N. Demassieux and W. Gehrke, "VLSI Architectures for Video Compression – A Survey," Proceedings of the IEEE, Vol. 83, No. 2, pp. 220-245, Feb. 1995.

[2] K. Kim, S. Jang and S. Kwon, "An Improvement of VLSI Architecture for 2-Dimensional Discrete Cosine Transform and Its Inverse," SPIE-Proceedings, Visual Communications and Image Processing ´96, Vol. 2727 Part 2, pp. 1017-1026, Mar. 1996.

[3] JC. Carlach, P. Penard and JL.Sicre, "TCAD: a 27 MHZ 8x8 Discrete Cosine Transform Chip," Proc. ICASSP, Vol. 4, pp. 2429-2432, May 1989.

[4] Rao and Yip, "Discrete Cosine Transform - Algorithms, Advantages, Applications," Academic Press, London 1990.

[5] R. M. Haralick, "A Storage Efficient Way to Implement the Discrete Cosine Transform," IEEE Trans. Computers, Vol. C25, pp. 764-765, Jul. 1976.

[6] M. Kovac and N. Ranganathan, "JAGUAR: A Fully Pipelined VLSI Architecture for JPEG Image Compression Standard," Proceedings of the IEEE, Vol. 83, No. 2, pp. 247-257, Feb. 1995.

[7] G.R. Goslin, "Using Xilinx FPGAs to Design Custom Digital Signal Processing Devices," Xilinx Inc., Appnotes 1998.

[8] ITU-T, "Recommendation H.263," Jul.5, 1995.

[9] Telenor Research, "VIDEO CODEC TEST MODEL, TMN5," http://www.nta.no/brukere/DVC/tmn5/tmn5.html, Jan. 31, 1995.

[10] K.-H. Noffz, R. Lay, R. Männer and B. Jähne, "Field Programmable Gate Array Image Processing," Handbook of Computer Vision and Applications, B. Jähne, H. Haußecker, P. Geißler (Editors), Vol. 3, Academic Press, 1999.

[11] Richard Schaphorst "Videoconferencing and Videotelephony – Technology and Standards," Artech House, Inc., 1996.

[12] William B. Pennebaker and Joan L. Mitchell, "JPEG, Still Image Data Compression Standard," Van Nostrand Reinhold, 1993.

[13] V. K. Madisetti and D. B. Williams, "The Digital Signal Processing Handbook," CRC Press LLC, 1998.

[14] Seehyun Kim and Wonyong Sung, "Finite Wordlength Effects Analysis and Wordlength Optimization of a Multiplier-Adder Based 8x8 D-IDCT Architecture," IEEE International Symposium on Circuits and Systems, Vol. 2, pp. 672-675, May 1996.

[15] S. Winograd, "On Computing the Discrete Fouriere Transform," Mathematics of Computation, Vol. 23, No. 141, pp. 175-199, 1978.

[16] Peter Alfke, "Efficient Shift Registers, LFSR Counters, and Long Pseudo-Random Sequence Generators," Xilinx XAPP 052, Ver. 1.1, Jul. 7, 1996.

[17] N. Camilleri and C. Lockhard, "Improving XC4000 Design Performance," Xilinx XAPP 043.

[18] E. Scopa, A. Leone, R. Ruerrieri and G. Baccarani, "A 2D-DCT Low-Power Architecture for H.261 Coders," Proc. ICAASP, Vol. 5, pp. 3271-3274, May 1995.

[19] Shin-ichi Uramoto, Yoshitsugu Inoue, Akihiko Takabatake, Jun Takeda, Yukihiro Yamashita, Hideyuki Terane and Masahiko Yoshimoto, "A 100-MHz 2-D Discrete Cosine Transform Core Processor," IEEE Jour. of Solid-State Circuits, Vol. 27, No. 4, pp. 492-498, April 1992.