A HIGH-PERFORMANCE HARDWARE IMPLEMENTATION OF THE H.264 SIMPLIFIED 8X8 TRANSFORMATION AND QUANTIZATION

Ihab Amer, Wael Badawy, and Graham Jullien

Advanced Technology Information Processing Systems (ATIPS)
Calgary, Alberta, Canada, T2N 1N4
{amer, badawy, jullien}@atips.ca

ABSTRACT

The recently approved digital video standard known as H.264 promises to be an excellent video format for use with a large range of applications. Real-time encoding/decoding is a main requirement for adoption of the standard to take place in the consumer marketplace. Transformation and quantization in H.264 are relatively less complex than their correspondences in other video standards. Nevertheless, for real-time operation, a speedup is required for such processes. Especially after the recent proposal to use an 8x8 integer approximation of Discrete Cosine Transform (DCT) to give significant compression performance at Standard Definition (SD) and High Definition (HD) resolutions. This paper discusses a high-performance hardware implementation of the H.264 simplified 8x8 transformation and quantization. The results show that the architecture satisfies the real-time constraints required by different digital video applications.

1. INTRODUCTION

Due to the remarkable progress in the development of products and services offering full-motion digital video, digital video coding currently has a significant economic impact on the computer, telecommunications, and imaging industry [1]. This raises the need for an industry standard for compressed video representation with extremely increased coding efficiency and enhanced robustness to network environments [2].

Since the early phases of the technology, international video coding standards have been the engines behind the commercial success of digital video compression. ITU-T H.264/MPEG-4 (Part 10) Advanced Video Coding (commonly referred as H.264/AVC) is the newest entry in the series of international video coding standards. It was developed by the Joint Video Team (JVT), which was formed to represent the cooperation between the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG) [3]-[5].

Compared to the currently existing standards, H.264 has many new features that makes it the most powerful and state-of-the-art standard [5]. Network friendliness and good video quality at high and low bit rates are two important features that distinguish H.264 from other standards [6]-[10].

Unlike current standards, the usual floating-point 8x8 DCT is not the basic transformation in H.264. Instead, a new transformation hierarchy is introduced that can be computed exactly in integer arithmetic. This eliminates any mismatch issues between the encoder and the decoder in the inverse transform [7], [11]. In the initial H.264 standard, which was completed in May 2003, the transformation is primarily 4x4 in shape, which helps reduce blocking and ringing artifacts.

In July 2004, a new amendment called the Fidelity Range Extensions (FRExt, Amendment I) was added to the H.264 standard. This amendment is currently receiving wide attention in the industry. It actually demonstrates further coding efficiency against current video coding standards, potentially by as much as 3:1 for some key applications. The FRExt project produced a suite of some new profiles collectively called High profiles. Beside supporting all features of the prior Main profile, all the High profiles support an adaptive transform-block size and perceptual quantization scaling matrices [5]. In fact, the concept of adaptive transform-block size has proven to be an efficient coding tool within H.264 video coding layer design [12]. This led to the proposal of a seamless integration of a new 8x8 integer approximation of DCT (and prediction modes) into the specification with the least possible amount of technical and syntactical changes [13]-[15].

So far, most of the work in H.264 is software oriented. However, a hardware implementation is desirable for consumer products to provide compactness, low power, robustness, cheap cost, and most importantly, real-time operation. In our previous work [16]-[26], we proposed hardware implementations for various blocks in the initial H.264 transformation hierarchy model and entropy coding. In this paper, we propose a high-performance hardware implementation of the H.264 newly-proposed simplified 8x8 transform and quantization.

The rest of this paper is organized as follows: Section 2 overviews the H.264 simplified 8x8 transform and quantization. In section 3, a description of the proposed hardware prototype is introduced. Section 4 presents the simulations and results achieved. Finally, section 5 concludes the paper.

2. H.264 SIMPLIFIED 8X8 TRANSFORM AND QUANTIZATION

An integer approximation of 8x8 DCT was proposed in FREXt to be added to the JVT specification based on the fact that at SD resolutions and above, the use of block sizes smaller than 8x8 is limited [15]. This transform is applied to each block in the luminance component of the input video stream. It allows for bit-exact implementation for all encoders and decoders. In spite of being more complex compared to the 4x4 DCT-like transform that is adopted by the initial H.264 specification, the proposed transform gives excellent compression performance when used for high-resolution video streams using a number of operations comparable to the number of operations required for the corresponding four 4x4 blocks using the fast butterfly implementation of the existing 4x4 transform [13], [14].

The 2-D forward 8x8 transform is computed in a separable way as a 1-D horizontal (row) transform followed by a 1-D vertical (column) transform as shown in Equation (1).

$$W = C_f X C_f^T \quad (1)$$

where the Matrix C_f is given by Equation (2).

$$C_f = \begin{bmatrix} 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 \\ 12 & 10 & 6 & 3 & -3 & -6 & -10 & -12 \\ 8 & 4 & -4 & -8 & -8 & -4 & 4 & 8 \\ 10 & -3 & -12 & -6 & 6 & 12 & 3 & -10 \\ 8 & -8 & -8 & 8 & 8 & -8 & -8 & 8 \\ 6 & -12 & 3 & 10 & -10 & -3 & 12 & -6 \\ 4 & -8 & 8 & -4 & -4 & 8 & -8 & 4 \\ 3 & -6 & 10 & -12 & 12 & -10 & 6 & -3 \end{bmatrix} .1/8 \quad (2)$$

Each of the 1-D transforms is computed using 3-stages fast butterfly operations as follows [14]:

Stage 1:

```
a[0] = x[0] + x[7];
a[1] = x[1] + x[6];
a[2] = x[2] + x[5];
a[3] = x[3] + x[4];
a[5] = x[0] - x[7];
a[6] = x[1] - x[6];
a[7] = x[2] - x[5];
a[8] = x[3] - x[4];
```

Stage 2:

```
b[0] = a[0] + a[3];
b[1] = a[1] + a[2];
b[2] = a[0] - a[3];
b[3] = a[1] - a[2];
b[4] = a[5] + a[6] + ((a[4]>>1) + a[4]);
b[5] = a[4] – a[7] – ((a[6]>>1) + a[6]);
b[6] = a[4] + a[7] – ((a[5]>>1) + a[5]);
b[7] = a[5] – a[6] + ((a[7]>>1) + a[7]);
```

Stage 3:

```
w[0] = b[0] + b[1];
w[1] = b[2] + (b[3]>>1);
w[2] = b[0] - b[1];
w[3] = (b[2]>>1) - b[3];
w[4] = b[4] + (b[7]>>2);
w[5] = b[5] + (b[6]>>2);
w[6] = b[6] – (b[5]>>2);
w[7] = -b[7] + (b[4]>>2);
```

Hence, the 2-D transform operation can be implemented using signed additions and right-shifts only, avoiding expensive multiplications. The post-scaling and quantization formulas are shown in Equations (3)-(5).

$$qbits = 15 + (QP\ DIV\ 6) \quad (3)$$

$$|Z_{ij}| = SHR(|W_{ij}|.MF + f, qbits + 1) \quad (4)$$

$$Sign(Z_{ij}) = Sign(W_{ij}) \quad (5)$$

where QP is a quantization parameter that enables the encoder to accurately and flexibly control the trade-off between bit rate and quality. It can take any integer value from 0 up to 51. Z_{ij} is an element in the quantized transform coefficients matrix. MF is a multiplication factor that depends on *(m = QP mod 6)* and the position *(i, j)* of the element in the matrix as shown in Table 1. *SHR()* is a procedure that right-shifts the result of its first argument a number of bits equal to its second argument. f is defined in the reference model software as $2^{qbits}/3$ for Intra blocks and $2^{qbits}/6$ for Inter blocks [3], [4].

Table 1. Multiplication Factor (*MF*)

m	(i, j) ∈ G_0	(i, j) ∈ G_1	(i, j) ∈ G_2	(i, j) ∈ G_3	(i, j) ∈ G_4	(i, j) ∈ G_5
0	13107	11428	20972	12222	16777	15481
1	11916	10826	19174	11058	14980	14290
2	10082	8943	15978	9675	12710	11985
3	9362	8228	14913	8931	11984	11295
4	8192	7346	13159	7740	10486	9777
5	7282	6428	11570	6830	9118	8640

*G_0: i = [0, 4], j = [0, 4]

G_1: i = [1, 3, 5, 7], j = [1, 3, 5, 7]

G_2: i = [2, 6], j = [2, 6]

G_3: (i = [0, 4], j = [1, 3, 5, 7]) ∩ (i = [1, 3, 5, 7], j = [0, 4])

G_4: (i = [0, 4], j = [2, 6]) ∩ (i = [2, 6], j = [0, 4])

G_5: (i = [2, 6], j = [1, 3, 5, 7]) ∩ (i = [1, 3, 5, 7], j = [2, 6])

3. HARDWARE PROTOTYPE

A block diagram of the proposed architecture is shown in Figure 1. The architecture uses 8x8 parallel blocks, QP, a synchronizing clock, and an enabling signal (Input Valid) as inputs. It outputs the quantized transform coefficients and the signal Output Valid. A block diagram of the architecture is shown in Figure 1.

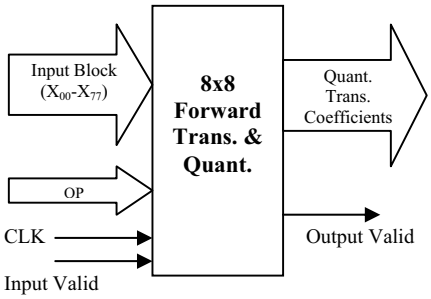

Figure 1. A block diagram of the proposed hardware architecture

The architecture is designed to perform pipelined operations, which drastically reduces the required memory resources and accesses, avoids any stall states, and dramatically improves the throughput of the architecture. Figure 2 gives a detailed block diagram of the proposed architecture showing the flow of signals between the main stages of the design.

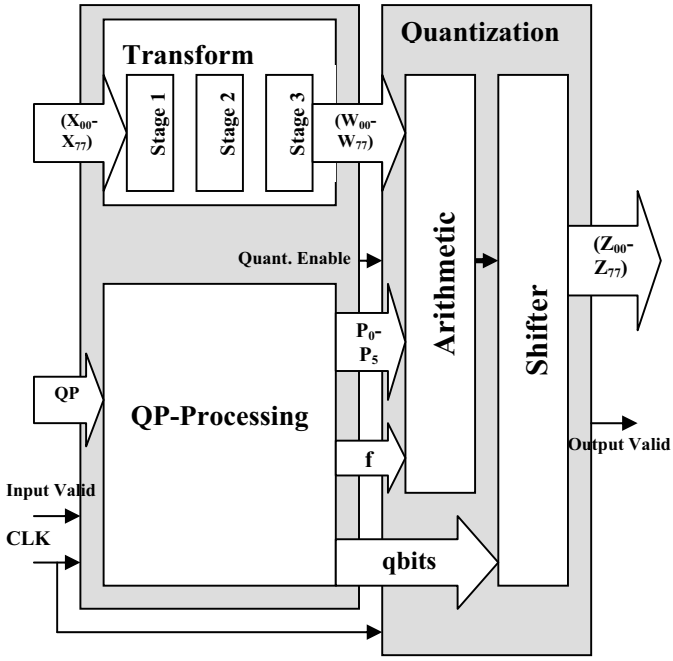

Figure 2. A detailed block diagram of the proposed hardware architecture

The architecture is composed of two main stages. The first one contains two blocks; the Transform block, which is composed of the three stages of the fast butterfly operations mentioned in Section 2, and the QP-Processing block, which is responsible for calculating the intermediate variables needed for quantization, such as *f*, *qbits*, and (P_0 – P_5), which are the values of the multiplication factors at the six different groups of positions in the matrix as shown in Table 1. Finally, the Quantization process takes place in the second main stage of the design. This is done by performing the addition and multiplication operations in the Arithmetic block, and finally the shifting operations in the Shifter block.

4. SIMULATIONS AND RESULTS

The architecture of the H.264 simplified 8x8 transformation is prototyped using VHDL language. It is simulated using the Mentor Graphics© ModelSim 5.4® simulation tool, and synthesized using Synplify Pro 7.1® from Synplicity©. The target technology is the FPGA device XC2V4000 (BF957 package) from the Virtex-II family of Xilinx©.

Table 2 summarizes the performance of the prototyped architecture.

Table 2. Performance of the prototyped architecture

Critical Path (ns)	CLK Freq. (MHz)	# of i/p Buffers	# of o/p Buffers
14.598	68.5	583	1217
# of I/O Reg. Bits	**#of Reg. Bits not inc. (I/O)**	**Total # of LUT**	**# of clock buffers**
1219	16893	29018	1

A 14.598 *ns* critical path is estimated by the synthesis tool. Since at steady state, the architecture outputs a whole 8x8 encoded block with each clock pulse, therefore the time required to encode a whole SD frame of 704 × 480 pixels can be calculated as follows:

$$\textit{Time required per CIF frame} = \textit{Time required per block} \times \textit{Number of blocks per frame}$$

$$= 14.598\ ns \times \frac{\textit{Number of pixels per frame}}{\textit{Number of pixels per block}}$$

$$= 14.598\ ns \times \frac{(704 \times 480)\ \textit{pixels per frame}}{(8 \times 8)\ \textit{pixels per block}}$$

$$\simeq 77.1\ \mu s$$

This value is about 216 times less than the 16.67 *ms* time required for continuous motion (assuming a refresh rate of 60 *frames/sec*). Similarly, it can be shown that the time required to encode a whole High Definition

Television (HDTV) frame of a 720×1280 pixels resolution, and a 60 *frames/sec* frame rate is 0.21 *ms*, which is about 79 times less than the 16.6 *ms* time required for continuous motion. Hence, the introduced architecture satisfies the real-time constraints for SD, HD, and even higher resolution video formats.

5. CONCLUSION

A high-performance architecture for prototyping the simplified 8x8 transformation and quantization, which was recently adopted by the H.264 standard is developed. The architecture is designed to perform pipelined operations, which drastically reduces the required memory resources and accesses, avoids any stall states, and dramatically improves the throughput of the architecture. It satisfies the real-time constraints required by different high-resolution digital video applications. The system is simulated using ModelSim 5.4®, and synthesized using Synplify Pro 7.1®, targeting the FPGA device XC2V4000 (BF957 package) from the Virtex-II family of Xilinx©. The ideal way to utilize the proposed design is to embed it to an overall system, where the reference software runs on a Digital Signal Processor (DSP), and the computationally extensive operations performed in the hardware block.

6. ACKNOWLEDGEMENT

The authors would like to thank Advanced Technology Information Processing Systems (ATIPS), Alberta Ingenuity Fund (AIF), Natural Science and Engineering Research Council of Canada (NSERC), Canadian Microelectronics Corporation (CMC), Micronet R&D, Alberta Informatics Circle of Research Excellence (iCORE), Canada Foundation for Innovation (CFI), and the Department of Electrical and Computer Engineering at the University of Calgary for supporting this research.

7. REFERENCES

[1] A. M. Tekalp, Digital Video Processing, Prentice-Hall, Inc., New Jersey, USA, 1995.

[2] "ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC," Draft Text of Final Draft International Standard for Advanced Video Coding, [Online]. Available: http://www.chiariglione.org/mpeg/working_documents.htm, March 2003.

[3] I. E. G. Richardson, "H.264/MPEG-4 Part 10: Transform & Quantization," A white paper. [Online]. Available: http://www.vcodex.com, March 2003.

[4] I. E. G. Richardson, H.264 and MPEG-4 Video Compression: Video Coding for Next-generation Multimedia, John Wiley & Sons Ltd., Sussex, England, December 2003.

[5] G. Sullivan, P. Topiwala, and A. Luthra, "The H.264 Advanced Video Coding Standard: Overview and Introduction to the Fidelity Range Extensions," SPIE Conference on Application of Digital Image Processing XXVII, Colorado, USA, August 2004.

[6] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC Video Coding Standard," IEEE Transactions on Circuits and Systems For Video Technology, Vol. 13, No. 7, pp. 560-576, July 2003.

[7] "Emerging H.264 Standard: Overview and TMS320DM642-Based Solutions for Real-Time Video Applications," A white paper. [Online]. Available: http://www.ubvideo.com, December 2002.

[8] K. Denolf, C. Blanch, G. Lafruit, and A. Bormans, "Initial memory complexity analysis of the AVC codec," IEEE Workshop on Signal Processing Systems, 2002 (SIPS'02), pp. 222-227, October 2002.

[9] T. Stockhammer, M. M. Hannuksela, T. Wiegand, "H.264/AVC in wireless environments," IEEE Transactions on Circuits and Systems For Video Technology, Vol. 13, No. 7, pp. 657-673, July 2003.

[10] M. Horowitz, A. Joch, F. Kossentini, A. Hallapuro, "H.264/AVC Baseline Profile Decoder Complexity Analysis," IEEE Transactions on Circuits and Systems For Video Technology, Vol. 13, No. 7, pp. 704-716, July 2003.

[11] H. S. Malvar, A. Hallapuro, M. Karczewicz, and L. Kerofsky, "Low-Complexity Transform and Quantization in H.264/AVC," IEEE Transactions on Circuits and Systems For Video Technology, Vol. 13, No. 7, pp. 598-603, July 2003.

[12] M. Wien, "Clean-up and improved design consistency for ABT," Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, doc. JVT –E025.

[13] S. Gordon, D. Marpe, and T. Wiegand, "Simplified Use of 8x8 Transform – Proposal," Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, doc. JVT –J029.

[14] S. Gordon, D. Marpe, and T. Wiegand, "Simplified Use of 8x8 Transform – Updated Proposal & Results," Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, doc. JVT –K028, Munich, Germany, March 2004.

[15] S. Gordon, "Simplified Use of 8x8 Transform," Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, doc. JVT –I022, San Diego, USA, September 2003.

[16] I. Amer, W. Badawy, and G. Jullien, "Towards MPEG-4 Part 10 System On Chip: A VLSI Prototype For Context-Based Adaptive Variable Length Coding (CAVLC)," accepted in *IEEE Workshop on Signal Processing Systems*, Austin, Texas, USA, October 2004.

[17] I. Amer, W. Badawy, and G. Jullien, "A VLSI Prototype for Hadamard Transform with Application to MPEG-4 Part 10," accepted in *IEEE International Conference on Multimedia and Expo*, Taipei, Taiwan, June 2004.

[18] I. Amer, W. Badawy, and G. Jullien, "Hardware Prototyping for The H.264 4x4 Transformation," proceedings of *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Montreal, Quebec, Canada, Vol. 5, pp. 77-80, May 2004.

[19] I. Amer, W. Badawy, and G. Jullien, "A SystemC Model for the MPEG-4 Part 10 4x4 DCT-like Transformation and Quantization," ISO/IEC JTC1/SC29/WG11 M10830, Redmond, USA, July 2004.

[20] I. Amer, W. Badawy, and G. Jullien, "A Hardware Block for the MPEG-4 Part 10 4x4 Transformation and Quantization," ISO/IEC JTC1/SC29/WG11 M10829, Redmond, USA, July 2004.

[21] I. Amer, W. Badawy, and G. Jullien, "A SystemC model for 4x4 Hadamard Transform and Quantization with application to MPEG-4 Part 10," ISO/IEC JTC1/SC29/WG11 M10828, Redmond, USA, July 2004.

[22] I. Amer, W. Badawy, and G. Jullien, "A Hardware Block for 4x4 Hadamard Transform and Quantization in MPEG-4 Part 10," ISO/IEC JTC1/SC29/WG11 M10827, Redmond, USA, July 2004.

[23] I. Amer, W. Badawy, and G. Jullien, "A SystemC model for 2x2 Hadamard Transform and Quantization with Application to MPEG–4 Part 10," ISO/IEC JTC1/SC29/WG11 M10826, Redmond, USA, July 04.

[24] I. Amer, W. Badawy, and G. Jullien, "A Hardware Block for 2x2 Hadamard Transform and Quantization with Application to MPEG–4 Part 10," ISO/IEC JTC1/SC29/WG11 M10825, Redmond, USA, July 04.

[25] I. Amer, W. Badawy, and G. Jullien, "An IP Block for MPEG-4 Part 10 Context-Based Adaptive Variable Length Coding (CAVLC)," ISO/IEC JTC1/SC29/WG11 M10824, Redmond, USA, July 2004.

[26] I. Amer, W. Badawy, and G. Jullien, "A Proposed Hardware Reference Model for Spatial Transformation and Quantization in H.264," accepted in Journal of Visual Communication and Image Representation Special Issue on Emerging H.264/AVC Video Coding Standard.