Efficient Hardware Architecture for EBCOT in JPEG 2000 Using a Feedback Loop from the Rate Controller to the Bit-Plane Coder

Grzegorz Pastuszak

Warsaw University of Technology, Institute of Radioelectronics,
Nowowiejska 15/19, 00-665 Warsaw, Poland
G.Pastuszak@ire.pw.edu.pl

Abstract. At the low compression ratio, the EBCOT engine of the JPEG 2000 encoder does not have to process all input data to achieve an optimal codestream in the sense of the rate-distortion criteria. This property is exploited in the architecture presented in this paper to allow higher throughputs of the JPEG 2000 encoder. An impact of the code block size and the internal FIFO size on the resultant speed is considered. The architecture is described in VHDL and synthesized for commercial FPGA technology. Simulation results show that at low compression ratios and for FPGA Stratix II devices, the single engine can support HDTV standards.

1 Introduction

The newest compression standards allow ever-higher compression ratio and support new functionality, although their computational complexity is still increasing. In image compression, JPEG 2000 [1], [2] incorporates some sophisticated algorithms, which require efficient implementation methods to shorten execution time. Embedded Block Coding with Optimized Truncation (EBCOT) is central to the standard and is a bottleneck of the codec. Usage of hardware acceleration makes it possible to obtain high throughputs.

There are some architectures and optimization methods for EBCOT presented in literature [3]-[7]. Also, some commercial solutions are available one the market, however, details are not published. Most implementation works for EBCOT have focused on the Bit-plane coder (BPC) and assumed that the Context Adaptive Binary Arithmetic Coder (CABAC) can process at most one binary symbol per clock cycle. We proved that it is possible to build efficient EBCOT architectures able to code two or three symbols per clock cycle [8]. Lossy compression gives an opportunity to shorten execution time since a number of input data do not contribute to the final JPEG 2000 codestream. There are a variety of design strategies exploiting this property. Some of them were proposed in [9], [10]. The main concept behind them is to skip less significant bit-planes of input coefficients based on rate-distortion criteria. This paper applies these approaches to the architecture able to code two symbols per clock cycle. Moreover, an impact of the code block size and the internal FIFO size on the resultant speed is considered.

F. Roli and S. Vitulano (Eds.): ICIAP 2005, LNCS 3617, pp. 604 – 611, 2005.

The rest of the paper is organized as follows: Section 2 reviews the EBCOT algorithm. Hardware acceleration methods and proposed design strategies for skipping of bit-planes are presented in Section 3. The architecture design is illustrated in Section 4. Section 5 gives evaluation results. Finally, Section 6 concludes the work.

2 EBCOT Algorithm

In the JPEG 2000 compression schema, input images undergo in turn: color transformation, wavelet transformation, and quantization. Each of these stages can be skipped depending on desired coding options. Quantized indices are grouped into rectangular structures (so-called code-blocks) and entropy-coded using the EBCOT algorithm.

2.1 Embedded Block Coding

The embedded block coder is known also as Tier-1 coder. The bit-plane coder is the first stage of the EBCOT algorithm. The BPC generates context-symbol pairs on the basis of quantization indices grouped in code-blocks. Input data are read in the sign-magnitude format and analyzed bit-plane wise starting from the most significant bit-plane (MSB) with a non-zero element to the least significant bit-plane (LSB). Each bit-plane is scanned in three coding passes called significance propagation, magnitude refinement, and cleanup.

Each pass provides a variable quality contribution to the reconstructed image. For the sake of the rate control algorithm, such an improvement should be calculated as a reduction in distortion, which may be obtained by summing reductions associated with each processed coefficient. For a single coefficient, the reduction in distortion can be calculated from bits located under the currently scanned bit-plane.

The context adaptive binary arithmetic coder (CABAC) is the second stage of the EBCOT algorithm. The CABAC module reads context-symbol pairs from the bit-plane coder and codes them into separate bit streams for each code-block. The CABAC contains the finite state machine that keep probability model for each context. The model identifies a binary value of the most probable symbol (MPS) and keeps an index pointing probability estimate of the least probable symbol (LPS).

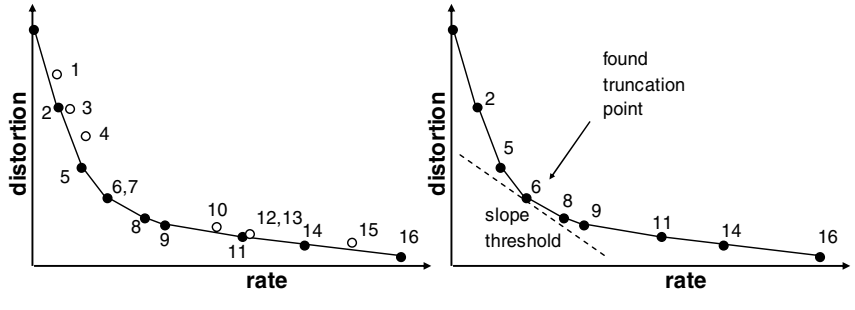

Fig. 1. Convex hull analysis and finding truncation point for a code-block

The main coding routine of the CABAC is based on the interval subdivision method of Shannon and Elias. The interval is described by its length and base. Successive renormalizations release bytes from MSB position of the base, incrementing the byte counter. Discrete truncation rates for each coding pass may be estimated on the basis of this counter increased by a small number (from 1 to 5) calculated from the internal variables. When the last pass is completed, the truncation length is equal to the number of released bytes.

2.2 Rate Control

The JPEG 2000 coder can employ Discrete Lagrange Optimization to achieve the target rate with a high accuracy. Inputs to the Discrete Lagrange Optimization are allowable truncation points described by reductions in distortion and rates, which are obtained from the BPC and the CABAC. The method finds the best truncation points for each code block to minimize the distortion of the reconstructed image subject to the target rate.

For each code block, the submitted points should form a convex hull to provide monotonic dependency of their slopes $\Delta D/\Delta R$. In order to meet this condition, some points have to be removed, as depicted in Fig. 1. The optimization procedure determines a global threshold with reference to the slopes. Next, a codestream for each code-block is truncated at the rate corresponding to the point having the smallest slope but greater than the global threshold. The final codestream including such truncated codestreams from each code block is optimal in the sense of the rate-distortion criteria. The JPEG 2000 encoder has to find the threshold that provides matching between the target and achieved rates.

3 Hardware Acceleration Methods

3.1 Embedded Block Coding

As each code-block is entropy-coded independently, the opportunity for parallel processing arises by using several block-coding engines. Speedup techniques for the BPC employ simultaneous scanning of one or more columns in a stripe and skipping no-operation samples. It reduces significantly local discontinuities of the output stream produced by the BPC unit. Another technique uses a FIFO buffer between the BPC and the CABAC. The reduction of time intervals, when input data for the CABAC are not available, depends on the FIFO size and the difference in speed between both main modules of the entropy coder. Most notably, the faster generation mitigates requirements for the capacity of the FIFO.

The main limitation on the throughput of EBCOT arises from the casual dependencies existing in the CABAC algorithm. Using a pipeline arrangement and parallel symbol encoding can increase the throughput.

3.2 Feedback Loop from the Rate Controller to the Bit-Plane Coder

Lossy compression gives an opportunity to shorten execution time since not all passes from a given code block contribute to the final JPEG 2000 codestream. To benefit

from this feature, the BPC should terminate processing of the code-block at the last pass included in the final codestream. In practice, this condition is difficult to evaluate since the optimal selection of last passes depends on the rate-distortion relations between all code-blocks in the image, as described in subsection 2.2. Hence, optimal truncation points can be found after all the code-blocks have been processed.

Final truncation points have rate-distortion slopes ($\Delta D/\Delta R$) greater than the threshold found in the Discrete Lagrange Optimization. An estimation of the threshold in parallel with processing of code-blocks allows the BPC to skip passes violating this condition. In [9], the estimation is accomplished by assigning the target rate to code-blocks processed so far. Thus, the value of the threshold increases during coding until the final value is found. Underestimation of the threshold adversely affects the efficiency of the early termination, e.g. the speedup is not as great as it would be. Owing to the temporal correlation in Motion JPEG 2000, the threshold can be taken from the preceding frame and modified with accordance to an adaptation rule [10].

An additional problem in embedding the feedback loop from the rate controller arises from the locally non-monotonic dependence between slopes corresponding to successive passes. This makes the correct detection of the termination condition difficult. For example, it may happen that the slope for a pass can fall below the temporal threshold validating the termination condition, but the slope of the next pass is greater and even merging these passes in the convex-hull analyzer makes the condition false. In this case, the termination would be performed too early.

The underestimation of the threshold cancels the negative impact of the too-early termination on the quality. The strength of these two factors should be balanced in order to prevent losses in quality and to minimize over-coding. This can involve additional conditions for the termination regarding the growth of the codestream length, the inclusion of zero-in-length cleanup passes, and the number of passes included in the collocated code-block in the preceding frame.

In hardware framework, the termination decision can be taken for a pass when both the reduction in distortion and the growth in the codestream length are known. The latter is determined with a delay caused by buffering (FIFO) and pipelining between the BPC and the rate estimator following the CABAC. As a consequence of the fact that the BPC continues processing until the termination, the delay has a similar effect on the coding time as the underestimation of the threshold. Varying the size of the FIFO changes the strength of this effect.

4 Architecture

4.1 Block-Coding Path

The BPC applies a pipeline arrangement and employs six memory modules to buffer quantized indices (2x1024x13 bits and 4x512x13 bits) and two memories to keep state variables (2x512 bits). There is a FIFO buffer as the last stage of the BPC. The bit-plane analysing method produces data at an average rate that outperforms by far the throughput of the CABAC able to code two context-symbol pairs per clock cycle. In particular, four columns in a stripe (16 bits) are scanned in one clock cycle. When there are more symbols to encode, additional clock cycles are inserted. To calculate

the reduction in distortion for each pass, bits from three bit-planes located just under the currently scanned bit-plane are read in parallel. If a coefficient is or becomes significant, its bits from all these bit-planes are mapped onto a value in the square error domain, and the result is added to a mean-square error (MSE) accumulator. At the end of each pass, the accumulator is flushed out.

The CABAC applies five pipeline stages. It is designed to process two context-symbol pairs per clock cycle [8]. The rate estimator, following the CABAC, calculates truncation rates on the basis of states of internal registers of the CABAC and generated code-bytes. The applied estimation of truncation rates is close to optimal because it discards code-bytes that are not necessary to decode correctly the last pass of a code-block.

4.2 Feedback Loop from the Rate Controller

The truncation rates along with the quality reduction, expressed as MSE, are forwarded from the block-coding path to the rate control one. The latter consists of a convex-hull analyzer, a feedback-threshold estimator, and the main rate controller.

The convex-hull analyzer embeds two small FIFO buffers to collect and adjust truncation rates and corresponding MSE reductions. A finite state machine (FSM) controls calculations of slopes and removes truncation points violating conditions on the convex hull. The convex-hull block incorporates a subcircuit converting inputs to their logarithmic representation. This removes the need to use the multiplication and division units and keeps the high dynamic range of slopes at 16 bits.

Positively classified truncation points are stored in a double-port memory interfacing the convex-hull analyzer with the feedback-threshold estimator. Both blocks access to two separate address spaces through their own ports. Exchange of address spaces allows communication. Hence, the blocks can operate simultaneously on successive code-blocks.

The feedback-threshold estimator provides the temporal threshold to the convex-hull analyzer, which in turn compares it with slopes calculated for successive truncation points of the currently-processed code-block. If a slope is less than the threshold, a termination signal is activated driving the BPC to finish coding at the end of the current pass. The feedback-threshold estimator handles a slope table accumulating rates. Each truncation point adds its rate to an accumulator addressed by the slope of this point. In the designed architecture, the threshold and slopes occupy 16 bits. To save hardware resources, eight most significant bits address the table. After updating the table by truncation points from a code-block, the temporal threshold is determined. The accumulators are read starting from the highest address. Their rates are accumulated in a global accumulator until its value is less than the total rate. The threshold is equal to the address of the table accumulator causing the violation of this condition. The architecture incorporates two slope tables accessed alternately on the image basis between the feedback threshold estimator and the Tier-2 coder.

All truncation points and code-bytes are forwarded to the external memory. When all code-blocks are coded, the Tier-2 of the JPEG 2000 encoder is activated. In particular, accurate truncation rates are determined, packet headers are coded, and final codestream is produced. Truncation rates are found on the basis of the accurate threshold by comparing it with the slope of each truncation point within each code-

block. Truncation rates are taken from the least significant points having slope greater than the threshold. The accurate threshold is determined in two steps. High order bits are obtained by reusing the slope table built by the feedback threshold estimator. Next, the table is reinitialized by truncation points (stored in the external memory) whose slopes have high-order bits equal to those of the threshold. Less significant bits are retrieved in the similar manner as higher ones.

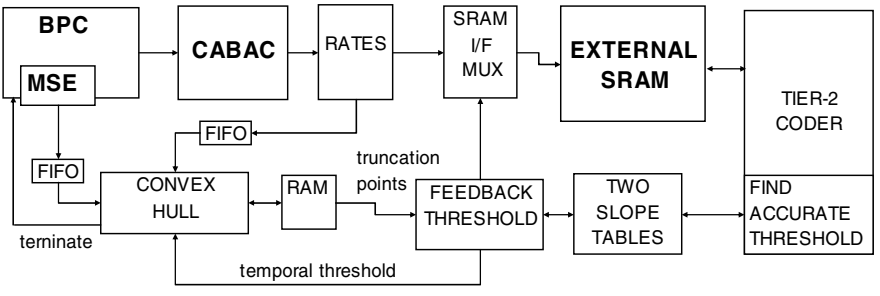

Fig. 2. Block diagram of the EBCOT engine

5 Implementation Results

The designed architecture of the EBCOT engine has been described in VHDL and verified against software reference model (JJ2000 version 5.1). Synthesis for the FPGA technology has been performed. The Tier-1 engine consumes 10 K Logic Elements and can operate at 120 MHz working frequency for FPGA Stratix-II devices. It enables the encoder to process about 40 million samples in the lossless mode (RCT, 5x3 wavelet filter core, no quantization). In the lossy mode, the throughput is higher and depends on applied quantization steps and the efficiency of the feedback loop from the rate controller.

Evaluations have been conducted for a set of images. Tables 1 and 2 show the number of clock cycles utilized to code the Baboon image by the block-coding engine enhanced or not by the feedback loop. The results for the Baboon image are presented in this paper since it involves the greatest deal of computations compared to other images. Evaluation conditions have been as follows: 512x512 grayscale image, ICT, 9x7 wavelet transform, two decomposition levels, quantization adjusted to the L2 norm of the wavelet filter. The tables show that the compression ratio, the size of the FIFO between the BPC and the CABAC, and the code-block size have an impact on the processing speed. In particular, lower compression ratio allows higher throughputs. Without the feedback loop, the number of clock cycles decreases with increasing the FIFO size. As discussed in subsection 3.2, this rule does not hold when exploiting the feedback loop. In this case, the number of clock cycles attains a minimum at some sizes of the FIFO, and it depends on the compression ratio, the code-block size, and the content of an image.

The tables compare two strategies of the threshold estimation. The first one approximates the temporal threshold by assignment of the total rate to code-blocks processed so far (increasing threshold). The second one applies the final threshold to all code-blocks), as this threshold would be taken from a preceding frame in a sequence (threshold from preceding frame). In this case, the reduction of clock cycles is the largest. When the threshold is underestimated, the savings in processing time are less. In some cases, the too-early termination can occur slightly deteriorating the quality of the reconstructed image. Nevertheless, these quality losses, if present, are very small (changes in PSNR are less than 0.05 dB) and can be neglected.

Table 1. Number of clock cycles necessary to code the Baboon image in the Tier-1 part for Code-block size 16x16

FIFO size x 32 bits		4	8	16	32	64
No feedback loop		807024	773855	738469	707850	707818
Increasing threshold	2 bpp	768179	739773	710901	698120	707124
	1 bpp	650985	633424	616659	615391	672120
	0.5 bpp	548867	536285	526335	529853	604490
Threshold from preceding frame	2 bpp	580200	568267	556982	560655	664117
	1 bpp	445103	434030	430286	434136	530044
	0.5 bpp	386522	378511	378001	386473	474921

Table 2. Number of clock cycles necessary to code the Baboon image in the Tier-1 part for Code-block size 64x64

FIFO size x 32 bits		32	64	128	256	512	1024
No feedback loop		848462	824180	796203	762422	736206	736157
Increasing threshold	2 bpp	791621	770341	753618	731745	717189	736154
	1 bpp	661254	644060	637568	617714	626650	686898
	0.5 bpp	540362	525738	516151	510806	520815	592612
Threshold from preceding frame	2 bpp	549247	541842	540840	530591	550585	664009
	1 bpp	386454	377120	367823	361692	368788	468429
	0.5 bpp	326829	316069	306247	296730	314427	399750

6 Conclusions

The architecture of the EBCOT with the feedback loop from the rate controller has been designed. It has been described in VHDL, verified, and synthesized for FPGA Stratix-II devices. The engine can operate at 120 MHz working frequency and can support HDTV standards at the low compression ratio.

Exploiting the feedback threshold from the rate controller improves the throughput of the EBCOT engine. The optimal selection of the FIFO size depends on the code-block size, the compression ratio. The temporal prediction of the threshold from the preceding frame gives the best speedup.

Acknowledgement

The work presented was developed within activities of VISNET, the European Network of Excellence, (http://www.visnet-noe.org), founded under the European Commission IST 6FP programme.

References

1. ISO/IEC 15444-1, Information technology - JPEG 2000 image coding system - Part I: Core Coding System, 2000
2. Taubman, D. S., Marcellin, M. W., JPEG2000: Image Compression Fundamentals, Standard and Practice. Norwell, MA: Kluwer, (2002)
3. Hsiao, Y.-T., Lin, H.-D., Lee, K.-B., Jen, C.-W.: High Speed Memory Saving Architecture for the Embedded Block Coding in JPEG 2000. (2002).
4. Andra, K., Chakrabarti, C., Acharya, T.: A high performance JPEG2000 architecture, IEEE Trans. Circuits and Systems for Video Technology, vol. 13, no. 3, pp. 209–218, (2003).
5. Lian, C.-J., Chen, K.-F., Chen, H.-H., Chen, L.-G.: Analysis and architecture design of block-coding engine for EBCOT in JPEG 2000, IEEE Trans. Circuits and Systems for Video Technology, vol. 13, no. 3, pp. 219–230, (2003).
6. Li, Y., Aly, R.E., Wilson, B., Bayoumi, M.A., Analysis and Enhancement for EBCOT in high speed JPEG 2000 Architectures, The 45th Midwest Symposium on Circuits and Systems, 2002. MWSCAS-2002., Volume: 2. (2002)
7. Fang, H.-C., Wang, T.-C., Lian, C.-J., Chang, T.-H., Chen, L.-G.: High Speed Memory Efficient EBCOT Architecture for JPEG2000, Proceedings of the 2003 International Symposium on Circuits and Systems, Vol. 2, pp 736-739, May 2003.
8. 8 Pastuszak, G.: A High-Performance architecture for arithmetic Coder in JPEG2000, ICME'04, Taipei, Taiwan, 2004.
9. Chang, T.-H., Lian, C.-J., Chen, H.-H., Chang, J.-Y., Chen, L.-G.: Effective Hardware-Oriented Technique For The Rate Control Of JPEG2000 Encoding. International Symposium on Circuits and Systems ISCAS 2003, Bangkok, Thailand. (2003)
10. Yu, W., Fritts, J., Fangting S.: Efficient Rate Control for Motion JPEG2000, Data Compression Conference (DCC 2004), Snowbird, UT, USA (2004)