A HARDWARE-ORIENTED ANALYSIS OF ARITHMETIC CODING – COMPARATIVE STUDY OF JPEG2000 AND H.264/AVC COMPRESSION STANDARDS

Grzegorz Pastuszak
Warsaw University of Technology, Institute of Radioelectronics, Nowowiejska 15/19, Warsaw, Poland

Keywords: CABAC, binary arithmetic coding, H.264, MPEG-4 AVC, JPEG 2000

Abstract: This paper provides an in-depth analysis and comparison of the arithmetic coding stages in the latest compression standards: JPEG 2000 and H.264/AVC for image and video systems, respectively. An impact of algorithm differences on hardware architecture is considered. Evaluation results show throughput requirements that real-time multimedia applications have to satisfy.

1 INTRODUCTION

Coding efficiency is the one of the most important features of all compression systems. Towards this goal, they follow a general schema of three main consecutive stages: modelling, quantization and coding. The latter stage exploits some well-known techniques, along with variable length codes and arithmetic coding, which map input symbols into binary sequences. The produced code streams achieve shorter lengths with respect to their source representation by making them dependent on occurrence probabilities, as Shannon's theorem claims. Arithmetic coders are able to attain better compression efficiency due to their property to effectively map input data onto binary sequences with fractional accuracy of lengths for entropy approximation. Adaptation to local statistics provides a path to further reduction of code stream lengths. However, these properties imply much higher computational complexity. JPEG 2000 (ISO/IEC 15444-1, 2000) and H.264/AVC (ISO/IEC 14496-10, 2003) are standards, where the Context Adaptive Binary Arithmetic Coding (CABAC) is the part of the compression scheme. Although CABAC bases on the same general principles in both standards, there are some substantial differences between them.

The comparison of the standards provided in (Marpe et. all, 2003-Oct.) focuses mainly on the compression efficiency providing complexity issues rather in broad outline. CABAC algorithms, as they stand, are extensively described in related works for both JPEG2000 (Taubman et. all, 2002) and H.264/AVC (Marpe et. all, 2003-July), (Marpe et. all, 2003-Sept.). In case of JPEG 2000, the bottleneck of the system arises from the entropy coding stage along with the arithmetic coder. Some optimisation methods were reported in literature and they lend themselves to the newest video compression schema. On the other hand, existing differences necessitate unique approaches. In this paper, we emphasize design details in terms of both hardware complexity and speed. Moreover, throughput evaluations are provided to find speed requirements for real-time applications.

The remainder of the paper is organized as follows: Section 2 illustrates consecutive stages of the CABAC that are deeply analysed in subsections 2.1 – 2.4. Subsection 2.5 addresses bypass mode variants. System-level conditions for arithmetic coding are given in Section 3.1. The following ones provide test conditions, evaluated requirements for processing speed and discussion combining them with hardware design methods; finally, Section 4 concludes the work.

2 MAIN STAGES OF THE CABAC

In terms of basic operations while executing the CABAC algorithm, we may distinguish four main stages. Fig. 1 shows their causal arrangement what suggest how to implement the algorithm in hardware

Pastuszak G. (2004).
A HARDWARE-ORIENTED ANALYSIS OF ARITHMETIC CODING – COMPARATIVE STUDY OF JPEG2000 AND H.264/AVC COMPRESSION STANDARDS.
In *Proceedings of the First International Conference on E-Business and Telecommunication Networks*, pages 309-316
DOI: 10.5220/0001386603090316

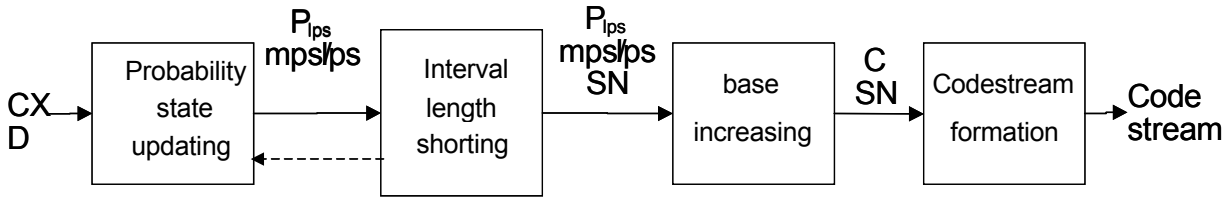

Figure 1: Division of the arithmetic coding algorithm in terms of casual relationships allows pipelined architectures. Here, Plps denotes probability estimate of LPS, SN – renormalization shift number, C – the base before renormalization, mps/lps – selects between MPS and LPS. The dashed line identifies optional feed back (JPEG2000).

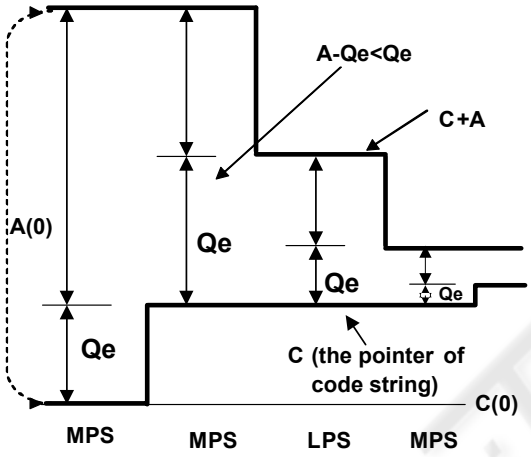

Figure 2: The interval subdivision in the JPEG2000 arithmetic coder.

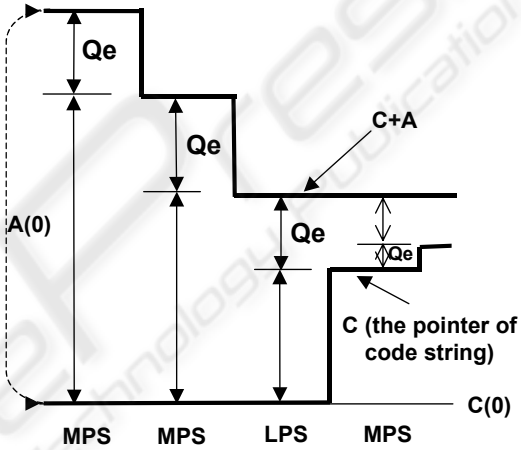

Figure 3: The interval subdivision in the H.264/AVC arithmetic coder.

to obtain a high throughput. First stage performs updating the probability state array according to pre-defined transaction rules. Each element of this state array corresponds to one of the possible input contexts, CX, and consists of two fields: the index and the most probable symbol (MPS) value. The index value identifies probability estimate of the least probable symbol (LPS). An estimate for a given context is forwarded to the second stage to subdivide the current probability interval (A) into two ones, as illustrated in Fig. 2 and Fig. 3. Depending on LPS/MPS coding, one of them is selected as a new one, and renormalized to desired range by shifting left, if needed. The third stage manages the interval base register (C - lower endpoint). This register is increased when the upper subinterval is selected as a new one. Successive renormalization shifts for the A register trigger the analogous behaviour of the C one, which releases code bits from its MSB positions. The bits are collected in the last stage into bytes and output to external functional blocks to form a final compressed stream.

2.1 Probability state updating

In JPEG 2000 and H.264/AVC, there are respectively 19 and 399 possible contexts defined for the CABAC. Each context has an associated finite state machine conveying the index, as a 6-bit vector, which assumes 47, in JPEG 2000, and 64, in H.264/AVC, allowable values. The small number of contexts, in the image compression standard, enables hardware architectures to implement the probability state array in registers, whereas the video schema imposes using an on-chip memory to save area of an integrated circuit. In spite of this drawback, H.264/AVC exhibits more flexible properties for pipeline-oriented approaches, since probability state updating process experience no impact from the probability interval renormalization. Such dependencies exist in the case of JPEG 2000, while coding MPS. If it does not cause renormalization shifts then the probability state, pointed by the current context, remains unchanged; otherwise a

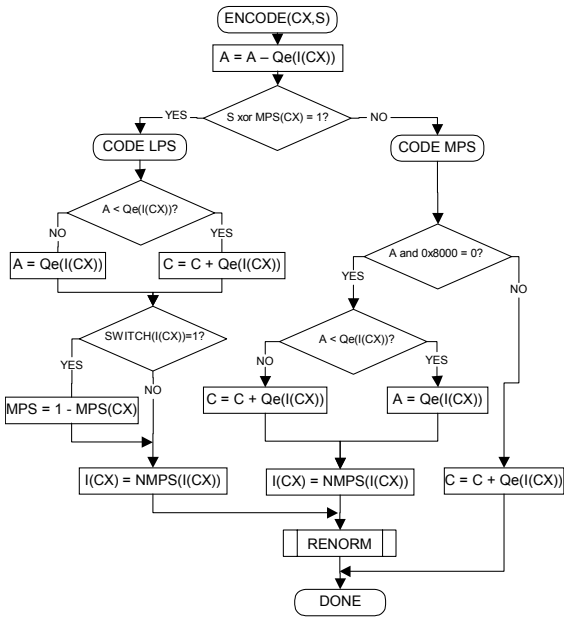

Figure 4: Arithmetic coding procedure in JPEG2000 embeds the conditional exchange of subintervals, conditional probability update for MPS.

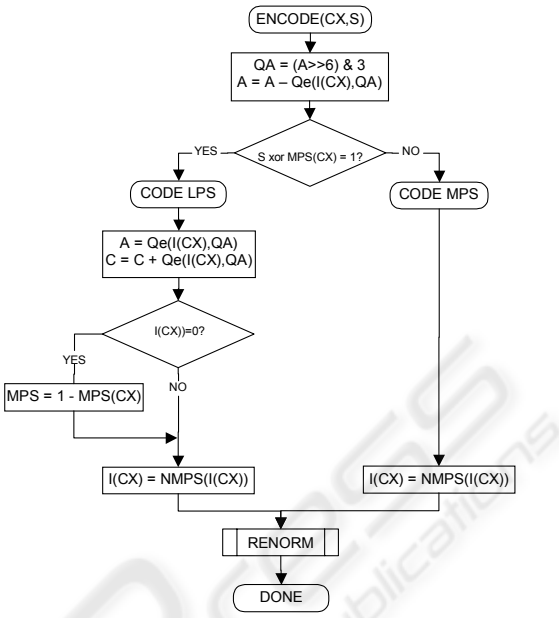

Figure 5: Arithmetic coding procedure in H.264/AVC: QA indicates two bits classifying the interval length (A) to one of four subranges to better match probability estimates. LPS is always assigned to upper subinterval, whereas MPS to the lower one.

new value is stored in accordance with the index transaction table. In the image compression schema, indices, in the probability state array, are initialised always to the same values defined in the standard specification. H.264/AVC involves quite complicated initialisation rules dependent on the quantization parameter (Qp), the frame type and a particular context number. Moreover, for INTER frames, we can select between three sets of initialisation schemas. The best choice may be done on the base of the resultant compression rate and depends on the video content. As for circuit, the initialisation rules introduce a considerable amount of both silicon and time resources. The first implication arises from the need to keep a large number of pre-defined constants in either the ROM table or combinatorial logic. The second one is caused by the necessity to check the rate for three initialisation cases, while coding INTER frames. Incorporating three CABAC engines, operating in parallel, can solve the time problem at the expense of hardware resources.

2.2 Interval length calculation

JPEG 2000 incorporates the 16-bits interval register, whereas H.264/AVC uses the 9-bits one. As for hardware, the increased precision improves slightly coding efficiency at the expense of resources and a longer carry chain. In considered standards, the interval length undergoes multiplication-free modifications, which realize its subdivision into two disjoint ranges. Since multiplication products are replaced by their tabled approximations, we would deal with some losses in coding efficiency (up to 3%). To mitigate this drawback, both compression schemes utilize different approaches. In JPEG 2000, the CABAC executes the conditional exchange, which ensures that the larger and smaller subintervals are always assigned to MPS and LPS, respectively. A fast hardware implementation translates this feature to a comparator (A<2Qe?) driving, together with MPS/LPS signal, the selection of an appropriate subinterval. The associated latency of this operation matches that in subtraction carry chain (A-Qe), thereby, the additional circuit should not deteriorate working frequency. H.264/AVC employs another solution to improve coding efficiency. Prior to subdivision (by subtraction), the interval is classified to one of four ranges based on its two bits located just after the MSB bit (which should equa1 1). Each range corresponds to a separate set of probability estimates to better approximate a multiplication product (A*Qe). Hence, given a set, the index points an appropriate estimate ready for either the subtraction or conditional interval reloading. A circuit, implementing the H.264/AVC arithmetic coder, has

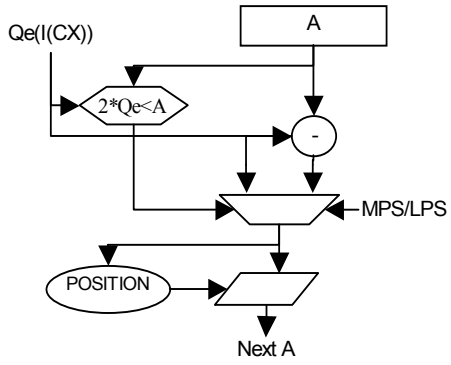

Figure 6: The A-interval subdivision circuit in the JPEG2000 arithmetic coder.

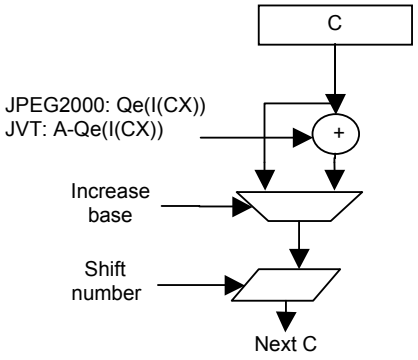

Figure 8: The C-base updating circuit able to process one symbol per clock cycle.

to provide all necessary probability estimate tables what amounts to a relevant combinatorial logic. The generated estimates should be available before beginning subinterval calculation to minimize propagation delays. Therefore, the pipelined architecture should determine these estimates in a preceding stage. However, selection between four ones, depended on the interval bits, must be held at the second stage what affects negatively timing. Determining estimates in advance fails in case of JPEG 2000 because of conditional state updating, as described above. On the other hand, there is no need for the selection. Moreover, less significant bits of estimates vary weakly over all indexes, what drives the logic synthesizer to simplify circuits, thus, shorting critical paths. After all, the interval value is renormalized by means of the shifting circuit. Fig. 6 and 7 depict interval length calculation circuits able to process one symbol per clock cycle for considered standards.

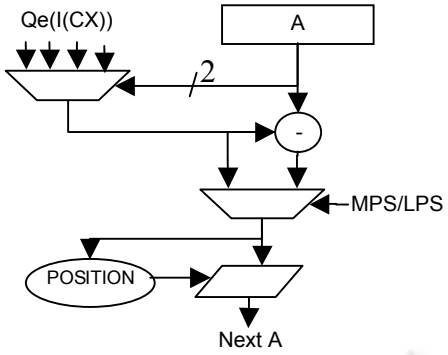

Figure 7: The A-interval subdivision circuit in the H.264/AVC arithmetic coder.

2.3 Interval base calculation

The base register in H.264/AVC needs 10 bits to be implemented. The location of subintervals is inverted with respect to JPEG 2000, i.e. in video compression scheme the upper and the lower correspond to LPS and MPS coding, respectively. Most notably, while subtracting the probability estimate from the interval length (LPS coding), the base has to be increased by the subtraction product. This dependency may affect negatively clock rate. Locating the base circuit in a separate pipeline stage gives somewhat shorter critical paths. As a consequence of similar bit counts in carry chains (9 and 10), the delays of pipelined circuits match one another. This observation holds for renormalization units due to an identical shift number submitted to both. The JPEG 2000 arithmetic coder keeps the lower bound of the interval in the 28 bits register. It experiences the analogous operations as the H.264/AVC counterpart. From hardware perspective, the main difference arises from their sizes what reflects on the latency in carry chains. In order to shorten critical paths in JPEG 2000, it is viable to divide the base register into two parts and place them in consecutive pipeline stages. Such rearrangement refers also to relevant combinatorial logic. Another distinguishing feature of the image compression coder lays in the meaning of the oldest part of the lower bound register, which supports carry propagation and code data releasing. The H.264/AVC version applies a different solution to these problems, so that, it removes the need to extend the base register towards MSB bits. The typical circuit managing the base is shown in Fig. 8.

2.4 Code stream generation

In the H.264/AVC arithmetic coder, output bits are released from the second MSB position of the base register after each single renormalization shift. In order to solve the problem of carry propagation, ones, encountered in series, are counted without outputting. Occurrence of the carry, indicated by the MSB bit of the base register, activates releasing binary one followed by a number of zeros. Otherwise, the inverted version of such sequence appears as an outcome after encountering a zero bit. This procedure requires the use of a counter signalling a total of outstanding bits. Its precision should match a maximal possible code stream length to prevent overflow when dealing with an extremely long series. The produced bits are assembled into bytes and released. It may happen that more then one byte has to be output due to a large number of outstanding bits. Provided the CABAC accepts one symbol per clock cycle, the design must adjust this rate to irregular code byte generation conditions by inserting wait-states. The JPEG 2000 arithmetic coder is free to that problem since at most two code bytes can appear as an outcome after processing one symbol. The algorithm imposes the need to keep the last generated byte in the buffer ready to complete carry. If there is the 0xFF byte, the control logic inserts one stuffing bit into the MSB position of the following byte. This bit assumes the zero value to trap a carry. A dedicated down-counter points to bits in the base register that have not been released so far. In terms of higher performances, both compression standards find a separate pipeline stage to make the code stream generation adequate.

2.5 Bypass mode

The CABAC in H.264/AVC provides the bypass mode, which, against the regular one, assumes uniform probability distribution of submitted symbols. Hence, it skips the probability state updating routine. Since related symbols contribute to the same code stream as in regular mode, it is natural to use the same resources with their timing constraints. The interval register remains unchanged in bypass mode. This property, in conjunction with skipping the probability adaptation, gives an opportunity to process bypassed symbols and regular ones in parallel. The probability estimates are obtained by single shifting right (division by 2) the interval value. Therefore, we must append one bit to represent estimate accurately. In case of JPEG 2000, the bypass mode forwards symbols directly to the output stream without arithmetic encoding. As for hardware, this approach allows increasing the throughput to a rate determined by the bit-plane coder performances, which submits input data to the CABAC module. However, the total improvement is not so significant since the standard enables the bypass mode for some coding passes over lowest bit planes.

3 EVALUATION

3.1 System-level constraints

Since the CABAC in H.264/AVC produces the single code stream for an entire slice, all necessary context-symbol data have to be applied to the one functional block. Thus, its speed determines the overall performances of the coder when input data are received continuously. Lower bit-rates decrease demands for throughput. Using rate-distortion optimisation for each macro-block improves quality at the same compression ratio. To obtain rates, we need to carry out arithmetic coding (when used) for all possible coding modes. As a consequence, it burdens the CABAC with a large number of computations and may lead to timing constraints for the encoder. JPEG 2000 supports entropy-coding parallelism by independent analysing rectangular blocks of coefficients in the wavelet domain. Each such code block generates a separate output stream, which can be truncated in some points to increase the compression ratio at the expense of quality losses of the reconstructed image. Moreover, a special mode drives the arithmetic coder to terminate the stream on these points. For the sake of the rate control policy, it is desired to produce more outcomes to discard their less significant parts with reference to the optimisation criteria. Thus, we need faster CABAC engines to support this property.

3.2 Evaluation conditions

Evaluations have been conducted for some video sequences taking into account the number of binary symbols submitted to the arithmetic coder in both standards. We examined test cases relating to CIF and QCIF resolutions. As reference software for image compression schema, we have employed JJ2000 version 5.1 adapted to support video material as Motion JPEG 2000 (ISO/IEC 15444-3, 2002). To get characteristics following options have been used: no tiling, five levels of wavelet decompositions, 9/7 wavelet filter, code block size of 64 x 64 samples, regular coding mode, single quality layer. Explicit quantization by step size has enabled to vary both

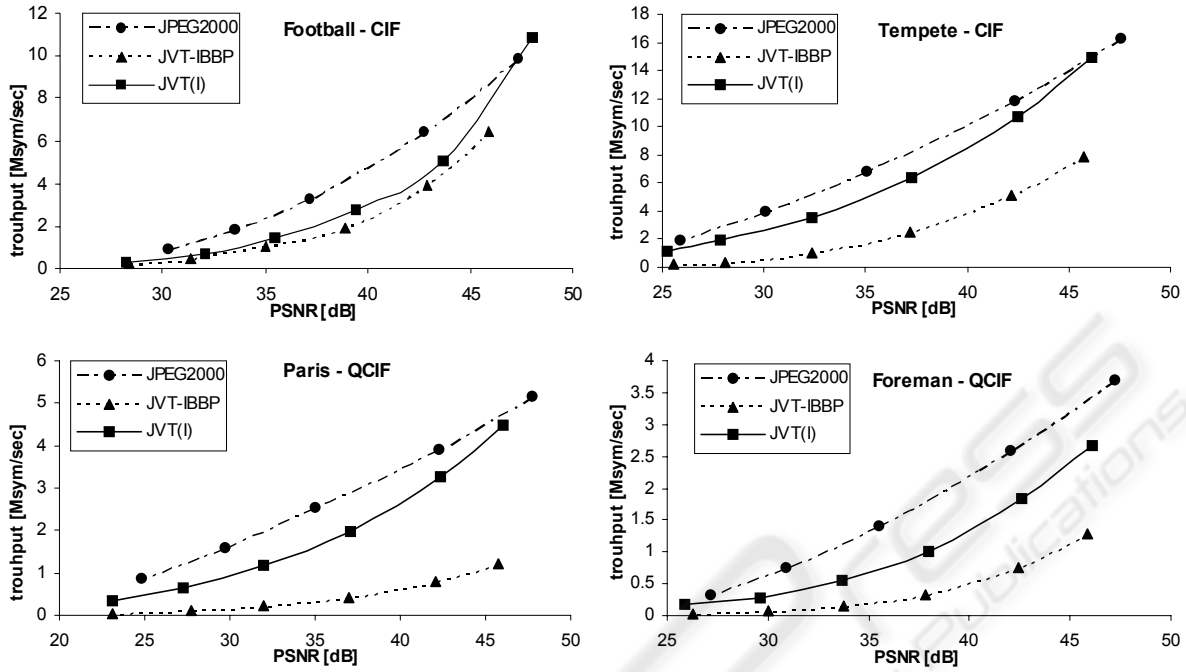

Figure 9. Averaged throughput requirements for H.264/AVC (JVT) and Motion JPEG2000

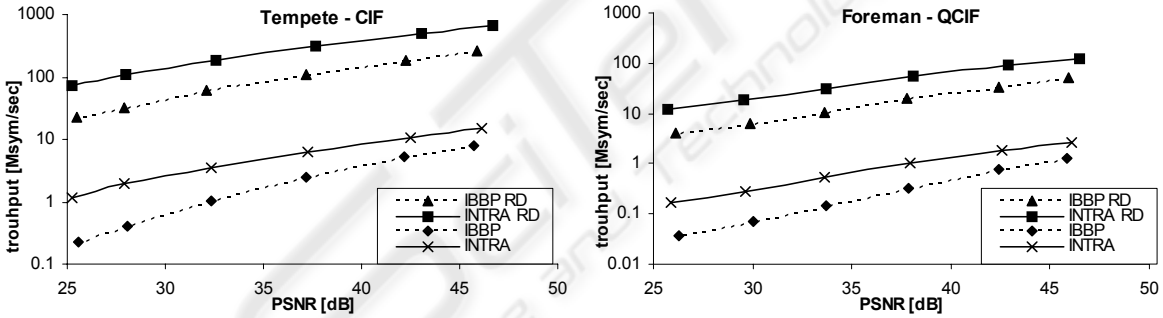

Figure 10. Averaged throughput requirements for H.264/AVC (JVT) using RD optimization or not.

quality and rate. Additionally, the special case, when no quantization is present, has been checked to demonstrate the most critical conditions, when total losses in quality of the reconstructed images results only from the rate-control policy. For encoding video material, the Joint Model (JM) of the Joint Video Team (JVT), software version 7.4 has been used. We have explored constraints when taking advantage of the RD-optimisation (rate-distortion) or not. All tests have skipped the rate controller to determine quality losses by explicit quantization parameter, Qp. The list of other settings has been as follows: only one slice per picture, 2 reference frames, full search of motion vectors, in IBBP mode: I-frames every 15th, 2 B-frames between I and/or P. All evaluations assumed frame rate of 30 Hz.

3.3 Evaluation results

Fig. 9 shows throughput (number of coded symbols) versus quality expressed as average PSNR (distortion) of the luminance component over all frames in a given sequence. The average PSNR of the chrominance components (U, V) has been adjusted with reference to that of the luminance one to obtain approximately the same differences for both compression schemas. This objective has been achieved by first producing curves for JPEG 2000, and then, iteratively varying the quantization parameter offset for chrominance components in H.264/AVC reference model. As expected, the throughput requirements increase significantly if we

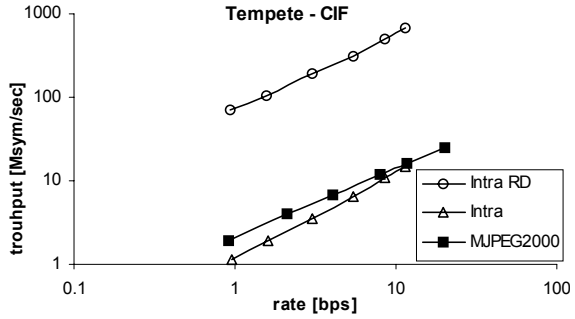

Figure 11. Averaged throughput requirements for H.264/AVC (JVT) and Motion JPEG2000 versus bit-rate.

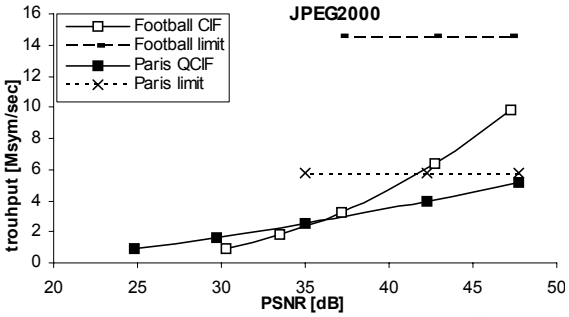

Figure 12. Averaged throughput requirements for Motion JPEG2000 with indicated maximal limits when no quantization.

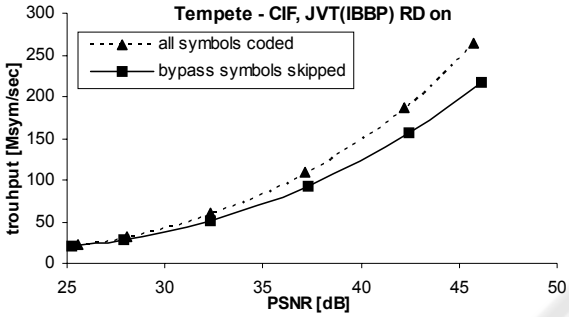

Figure 13. Averaged throughput requirements for H.264/AVC diminish when symbols obeying uniform distribution are skipped.

want to decrease distortions. Without RD optimisation, the video compression schema takes less time to accomplish the arithmetic coding routine than JPEG 2000. The INTRA mode of H.264/AVC demands a little smaller throughput then JPEG 2000 for the quality range of interest to most video applications. However, at higher qualities, both standards need similar processing speed of the CABAC block with the tendency to favour the JPEG 2000. Of course, the use of INTER mode makes H.264/AVC the best solution in terms of both compression ratio and throughput of the entropy coder. On the other hand, the compressed stream must embed the INTRA frames, so, their temporal impact on the latency of the arithmetic coder should be taken into account. Using RD optimisation in H.264/AVC increases demands for the CABAC module by about two orders of magnitude (see Fig. 10). This computation growth is necessary only to select the best mode for macroblock in the sense of Lagrange's minimization. The opportunity for parallelism arises, since a macroblock can be processed for different modes at the same time. As depicted in Fig. 11, the computation demands for the CABAC engine are almost proportional to a given rate in both standards. So, on the base of imposed bandwidth or storage limitations, we can estimate desired throughput of the arithmetic coder.

3.4 Discussion

With the above observations, we re-examine the pipelined structure of the arithmetic coders for real time performances. Since the context generator in JPEG 2000 occupies a relatively large silicon area of the integrated circuits (due to coefficient memory), it is payable to optimize the throughput of the CABAC unit rather then to duplicate entropy coding paths including both modules. Such single path, embedding pipeline architecture able to process one symbol per clock cycle, yields speed to target from CIF sequences (4:2:0 – 4:4:4, 30 frames per second) provided regular and lossless mode (Hsiao et. all, 2002), (Lian et. all, 2003), (Li et. all, 2002), (Fang et. all, 2002). Approaches based on sequential arrangement attain worse results in spite of higher clock rates (Andra K et. all, 2003). The lossy compression allows higher throughputs, as shown in Fig. 12. Of course, the exact performances depend primarily on the technology of an integrated circuits and efforts spent to minimize critical paths. To speed up the entropy coding, we can use parallel processing paths and/or modify the architecture to process two or more symbols per clock cycle.

In H.264/AVC encoder without RD optimisations, the single CABAC engine, complying with pipeline arrangement (like in JPEG 2000) able to process one symbol per clock cycle, can easy support PAL and NTSC standards in the compression range of most interest. Moreover, it makes possible to target HDTV at lower bit-rates (low quality). However, we must remember that the throughput of the whole video coder depends on the motion estimation unit rather then the entropy coding stage. Taking

advantage of the RD optimisation finds the arithmetic coder to become another bottleneck of the system. As mentioned above, employing several engines in parallel mitigates timing constraints at the cost of silicon resources. This can be realized in two ways. The first assumes dividing a frame into a number of slices and assigning one CABAC unit to each of them. Prior to checking all coding scenarios, we have to save the states of internal registers, and then, encode a macroblock in series for various modes staring from the same state. The second way determines rates simultaneously in separate coding units. One can combine these approaches as well. Since some symbols obey uniform distribution, they induce extension of the output stream by one bit. So, when we want to estimate rates, it is enough to count them without submitting to the arithmetic coder. Fig. 13 depicts the difference in throughput of the CABAC, while benefiting from this opportunity.

4 CONCLUSIONS

The analysed arithmetic coding algorithms proves to comply with the general schema of the pipeline architecture design. Corresponding stages exhibit some variants of the CABAC concepts requiring different approaches to minimize critical paths. The H.264/AVC version can achieve higher working frequencies than the JPEG 2000 one due to smaller sizes of the key registers. Owing to the latter supports entropy coding parallelism, it can achieve high performance, but a hardware designer should primarily sophisticate the single entropy channel to save a total of silicon area. Special attention must be paid to optimise the CABAC unit in H.264/AVC, when RD optimisation is on, including parallel encoding engines, counting bypassed symbols, and minimizing critical paths. Without RD enhancements, the throughput of the single CABAC gives opportunity even to target HDTV.

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

ISO/IEC 15444-1, 2000, *JPEG 2000 Part I Final Committee Draft Version 1.0*.

ISO/IEC 15444-3, 2002, *Motion – JPEG 2000 Part 3*.

ISO/IEC 14496-10, 2003, *ITU-T Recommendation H.264 and ISO/IEC 14496-10 MPEG-4 Part 10, Advanced Video Coding (AVC)*,.

Taubman D. S. and Marcellin M. W., 2002, *JPEG2000: Image Compression Fundamentals, Standard and Practice*. Norwell, MA: Kluwer.

Andra K., Chakrabarti C., and Acharya T., 2003, "A high performance JPEG2000 architecture," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 13, no. 3, pp. 209–218.

Hsiao Y-T, Lin H-D, Lee K-B and Jen C-W, 2002, "High Speed Memory Saving Architecture for the Embedded Block Coding in JPEG 2000".

Lian C.-J., Chen K.-F., Chen H.-H., and Chen L.-G., 2003, "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.

Li Y., Aly R.E., Wilso B. and Bayoumi *M.A.*, 2002, "Analysis and Enhancement for EBCOT in high speed JPEG 2000 Architectures," The 45th Midwest Symposium on Circuits and Systems, 2002. MWSCAS-2002.

Fang H-C, Wang T-C, Lian C-J, Chang T-H and Chen L-G, 2002, "High Speed Memory Efficient EBCOT Architecture for JPEG2000".

Marpe D., Schwarz H, and Wiegand T., 2003-July, "Context-Based Adaptive Binary Arithmetic Coding in the H.264/AVC Video Compression Standard", *IEEE Transactions on Circuits and Systems for Video Technology*.

Marpe D. and Wiegand T., 2003-Sept., "A highly Efficient Multiplication-Free Binary Arithmetic Coder and Its Application in Video Coding", *Proc. IEEE International Conference on Image Processing (ICIP 2003)*, Barcelona, Spain.

Marpe D., George V., Cycon H.L., Barthel K.U., 2003-Oct, "Performance evaluation of Motion-JPEG2000 in comparison with H.264/AVC operated in pure intra coding mode", *Proc. SPIE on Wavelet Applications in Industrial Processing*, Rhode Island, USA.