A HIGHLY EFFICIENT MULTIPLICATION-FREE BINARY ARITHMETIC CODER AND ITS APPLICATION IN VIDEO CODING

Detlev Marpe and Thomas Wiegand

Fraunhofer Institute for Communications – Heinrich Hertz Institute, Image Processing Department
Einsteinufer 37, D-10587 Berlin, Germany, [marpe,wiegand]@hhi.fhg.de

ABSTRACT

A novel and highly efficient algorithm of multiplication-free binary arithmetic coding is proposed. Our proposed method relies on simple table lookups for performing the computationally critical operations of interval subdivision and probability estimation. Moreover, the underlying design principle provides a great flexibility for serving the different needs of all kind of coding applications where binary or binarized data have to be processed. A binary arithmetic coder of the type described in this paper has become part of the CABAC entropy coding scheme of the emerging H.264/AVC video coding standard. Experiments using this binary arithmetic coder in its native video coding environment demonstrate a superior coding efficiency as well as a significantly higher throughput rate in comparison to the MQ coder, which is currently being considered state-of-the-art in fast binary arithmetic coding.

1. INTRODUCTION

Arithmetic coding has attracted a growing attention in the past years. Recently developed image coding standards like JBIG-2, JPEG-LS or JPEG2000 [6], and the emerging video standard H.264/AVC [8] all include arithmetic coding, at least as an optional feature. As a common requirement of all these applications, there is the need for a fast and efficient (binary) arithmetic coding scheme. Efficient implementations of arithmetic coding both in hardware and software, however, are not only required in the application domains of image and video coding; general bit-based data compression schemes and advanced methods of text compression need high throughputs of arithmetic codes as well [3].

Binary arithmetic coding is based on the principle of recursive *interval subdivision* that involves the following elementary multiplication operation. Suppose that an estimate of the probability p_{LPS} of the *least probable symbol* (*LPS*) is given and that the given coding interval is represented by its lower bound (base) L and its width (range) R. Based on that settings, the given interval is subdivided into two sub-intervals: one interval of width

$$R_{LPS} = R \times p_{LPS}, \qquad (1)$$

which is associated with the LPS, and the dual interval of width $R_{MPS} = R - R_{LPS}$, which is assigned to the *most probable symbol* (*MPS*) related to a probability estimate of $1 - p_{LPS}$. Depending on the observed binary decision, either identified as the LPS or the MPS, the corresponding sub-interval is then chosen as the new coding interval. A binary value pointing into that interval represents the sequence of binary decisions processed so far, while the range of that interval corresponds to the product of the probabilities of those binary symbols. Thus, to unambiguously identify the coding interval and hence the coded sequence of binary decisions, the Shannon lower bound on the entropy of the sequence is asymptotically approximated by using the minimum precision of bits specifying the lower bound of the final interval.

In a practical implementation of binary arithmetic coding the main bottleneck in terms of throughput is the multiplication operation in Eq. (1) required to perform the interval subdivision. A significant amount of work has been published in literature aimed at speeding up the required calculation in (1) by introducing some approximations of either the range R or of the probability p_{LPS} such that the multiplication can be avoided [1]–[3]. Among these low-complexity binary arithmetic coding methods, the Q coder [1] and its derivatives QM and MQ coder [6] have been most successfully applied, especially in the context of the still image coding standardization groups JPEG and JBIG.

In this paper, we present a new and flexible design of a multiplication-free binary arithmetic coder, the so-called *modulo coder* (*M coder*). Actually, the proposed M coder can be associated with a whole family of binary arithmetic coders, one of which is representing the coding engine of the video standard H.264/AVC [8]. We will show simulation results demonstrating the superior performance of the proposed design in comparison to the state-of-the-art MQ coder both in terms of coding efficiency and throughput rate.

2. PROPOSED METHOD

The basic idea of our multiplication-free approach to binary arithmetic coding is to project both the legal range $[R_{\min}, R_{\max})$ of the interval width R and the range

0-7803-7750-8/03/$17.00 ©2003 IEEE

$[p_{min}, 0.5]$ of probabilities associated with the LPS onto a small set of representative values $\boldsymbol{Q} = \{Q_0, \ldots, Q_{K-1}\}$ and $\boldsymbol{P} = \{p_0, \ldots, p_{N-1}\}$, respectively. By doing so, the multiplication on the right hand side of (1) can be approximated by using a table of $K \times N$ pre-computed product values $Q_k \times p_n$ with $0 \le k \le K-1$ and $0 \le n \le N-1$.

For the applicability of our method, the following conditions are assumed to be fulfilled:[1]

(i) A renormalization strategy equivalent to that introduced in [4] is applied to R such that R is forced to stay within the interval $[2^{b-2}, 2^{b-1})$ for a given b-bit implementation.

(ii) The lower bound of the LPS related probability is given by $p_{min} \ge 2^{-(b-2)}$ to prevent underflow of the arithmetic in (1).

(iii) The probability estimator is realized by a Markov chain, which allows a table-driven implementation of the adaptive probability estimation [5]. We further assume that the LPS related probabilities are monotonically decreasing with increasing state index n, i.e., $0.5 = p_0 \ge \ldots \ge p_n \ge \ldots \ge p_{N-1} = p_{min}$, and that the state transition rules are given by at most two tables *TransStateLPS* and *TransStateMPS* each having N entries and pointing to the next state for a given entry of a state n with $0 \le n \le N-1$ and for a observation of a bit identified either as a LPS or MPS, respectively.

(iv) The legal range interval $[R_{min}, R_{max}) = [2^{b-2}, 2^{b-1})$ induced by condition (i) is uniformly quantized into $K = 2^{\kappa}$ cells, where $1 \le \kappa \le (b-2)$ and the cells are indexed according to their increasing reconstruction values $Q_0 < \ldots < Q_k < \ldots < Q_{K-1}$.

Let *TabRangeLPS* denote the 2-D table containing the $K \times N$ product values $Q_k \times p_n = TabRangeLPS[\, n, k \,]$. Then, condition (i) and (iii) imply that it is sufficient to supply the elements of *TabRangeLPS* in $(b-2)$-bit precision resulting in a memory requirement of $2^{\kappa} \; N \; (b-2)$ bits for the whole table.

2.1. Interval subdivision

Under the conditions (i) – (iv), we assume the current LPS related probability estimate p_{LPS} to be (approximately) represented by its corresponding state index n, and the current coding interval to be specified by its lower bound L and range R. Then, the interval subdivision operation in (1) will be approximated in the proposed M coder by a simple table look-up operation

$$R_{LPS} = TabRangeLPS[\, n, k(R) \,], \tag{2}$$

where the (quantization) cell index k can be calculated as the combination of a bit shift and a bit masking operation:

$$k(R) = (R >> q) \;\&\; (2^{\kappa} - 1). \tag{3}$$

The bit shift operand in (3) is defined by $q = b-2-\kappa$ and the bit masking operation in (3) can be interpreted as a *modulo operation* using the operand $K = 2^{\kappa}$, which motivates the naming of our proposed coder.

Firstly, it is important to note that unlike e.g. in the MQ coder, there is no need to tabulate the representative *LPS* probability values $\{p_n \mid 0 \le n \le N-1\}$ in the M coder, since each probability value p_n is only implicitly addressed by its corresponding state index n in the interval subdivision of (2) as well as in the probability state transition process. Secondly, instead of using the representative quantized range values $Q_0, \ldots, Q_{K-1}$ explicitly in the M coder, the quantized range value of a given value R is only addressed by its quantizer index $k(R)$, as given in Eq. (3). Thirdly, for a given probability estimator, the accuracy of the approximation in Eq. (2) depends on the granularity of the uniform quantization of the range interval $[R_{min}, R_{max}) = [2^{b-2}, 2^{b-1})$, which, in turn, is determined by the parameter κ with $1 \le \kappa \le (b-2)$. In a practical scenario, the most reasonable approach for a suitable selection of the parameter κ is to jointly optimize both the probability estimator and the parameter κ under some pre-defined constraints regarding the memory consumption or/and the loss in coding efficiency.

To simplify matters, we restrict our further analysis to a special kind of probability estimator as presented in the next section.

2.2. Probability estimation

Given the condition (iii), we recursively define the following set of representative LPS probability states $\boldsymbol{P} = \{p_0, \ldots, p_{N-1}\}$:

$$p_i = \alpha \cdot p_{i-1} \quad \text{for all } i = 1, \ldots, N-1$$
$$\text{with } \alpha = \left(\frac{p_{min}}{0.5}\right)^{1/N-1} \text{ and } p_0 = 0.5. \tag{4}$$

The associated transition rules for the *LPS* probability are based on the following relation between a given *LPS* probability p_{old} and its updated counterpart p_{new}:

$$p_{new} = \begin{cases} \max(\alpha \cdot p_{old}, p_{min}), & \text{if a } MPS \text{ occurs} \\ \alpha \cdot p_{old} + (1-\alpha), & \text{if a } LPS \text{ occurs} \end{cases} \tag{5}$$

where the value of α is given as in (4). This kind of estimator has been proposed in [3]. The basic idea behind this design is that, on the one hand, for LPS probabilities near 0.5, there is no need for an accurate estimation, since for two closely neighboring states nearly the same code length is achieved. On the other hand, if the LPS probability gets close to 0, or equivalently, the MPS probability approaches 1, a more accurate representation of the probabilities is needed to achieve a near-optimal code length in the subsequent arithmetic coding process.

[1] Some of these conditions can be further relaxed at the expense of the clarity of presentation.

With regard to a practical implementation of the probability estimator defined by Eqs. (4) and (5), it is important to note that all transition rules can be realized by at most two tables each having N entries. Thus, this type of estimator is fully compliant with condition (iii) above. Actually, for this probability estimator it is sufficient to provide a single table *TransStateLPS*, which determines for a given state index n the new updated state index *TransStateLPS*[n] in case a *LPS* has been observed. The *MPS* driven transitions can be simply obtained by (saturated) increments of a given state index n by the fixed value of 1 resulting in an updated state index min (n+1, N–1).

2.3. Design considerations

In designing a probability estimator of the above given type, a choice of the scaling factor α and the cardinality N of the set of LPS probabilities to be modeled must be made. In general, the actual choice should represent a good compromise between the desire for fast adaptation ($\alpha \to 0$; small N), on the one hand, and the need for a sufficiently stable and accurate estimate ($\alpha \to 1$; larger N), on the other hand. Instead of using α as the free parameter, it is often more useful to determine p_{min} according to some *a priori* knowledge of the statistical properties of the data to process and to adjust α according to the chosen p_{min} and N. For the choice of N, usually additional constraints on the size 2^{κ} N (b–2) of the *TabRangeLPS* table are applied.

Figure 1 shows the results of an experimental evaluation of different choices for $K = 2^{\kappa}$ and N under the constraint of a fixed table size of 256 for *TabRangeLPS*, where $b = 10$ was chosen as the numerical implementation precision. The underlying coding experiments were performed by using the M coder in its native CABAC environment of the H.264/AVC video coder. The dashed line in the graph of Fig. 1 illustrates the different possible combinations of pairs (K, N) with constant product K $N = 256$. Only a discrete number of combinations corresponding to the values $\kappa = 1$, 2 and 3 were actually tested and the pre-defined value of $p_{min} = 0.01875$ in H.264/AVC was kept fixed.

For each legal combination (K, N), a coding experiment was performed by evaluating a test set of 4 sequences in QCIF resolution and 3 sequences in CIF resolution at different quantization parameters QP. The corresponding averaged and interpolated bit-rate savings relative to the basic configuration of (K, N) = (2, 32) are depicted by the solid line in Fig. 1. As can be seen from the graph, a distinguished maximum averaged bit-rate saving of 1.1% has been obtained for the choice of $\kappa = 2$ and $N = 64$. This choice represents the configuration of the M coder as specified in the H.264/AVC standard [8], where the scaling factor related to this choice of $N = 64$ and $p_{min} = 0.01875$ is given by $\alpha \approx 0.95$. In comparison to this

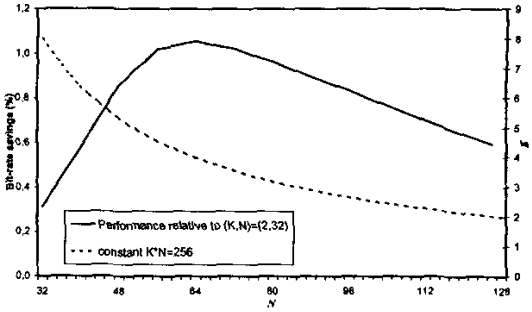

Figure 1: Averaged bit-rate savings (solid line) for different choices of (K, N) with constant K $N = 256$ (dashed line) relative to the basic configuration of (K, N) = (2, 32).

configuration of H.264/AVC, smaller gains were observed for smaller values of N and larger values of K, whereas for larger values of N and smaller values of K, gains in the range of 0.6–1.0% relative to the basic configuration of (K, N) = (2, 32) were obtained. Overall, however, the performance differences between the tested M coder configurations were rather small.

2.4. Bypass coding mode

To speedup encoding (and decoding) of symbols, for which $R - R_{LPS} \approx R_{MPS} \approx R/2$ is assumed to hold, i.e., for symbols that are assumed to be nearly uniformly distributed, a complete "bypass" of the probability estimation and update process is proposed for the M coder. According to the assumption, the corresponding interval subdivision operation can be simplified in such a way that an equipartition of the coding interval is achieved without the explicit table look-up operation of Eq. (2). However, instead of halving the current interval range R, the base L of the coding interval is doubled before choosing the lower or upper sub-interval depending on the value of the symbol to encode (0 or 1, respectively). In this way, the coding process is further simplified since doubling of L and R in the subsequent renormalization is no longer required provided that the renormalization in bypass coding mode is operated with doubled decision thresholds.

3. SIMULATION RESULTS

Two sets of experiments were conducted to evaluate both the coding efficiency and the throughput of the M coder in comparison to the MQ coder. For that purpose, a fairly optimized implementation of the MQ coder as part of the UBC Power-JBIG-2 implementation [9] has been integrated both into the test model of H.264/AVC [8], software version JM50c, and into a real-time software-based H.264 decoder implementation. The initialization of the CABAC context models has been modified such that the initial states corresponding to the CABAC engine have been mapped to their corresponding closest states of the MQ coder.

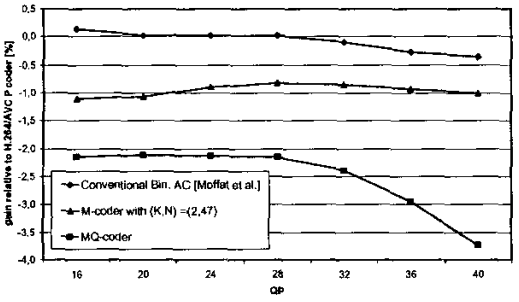

Figure 2: Averaged bit-rate savings vs. quantization parameter QP for different binary arithmetic coders in H.264/AVC (negative values correspond to an increase in bit-rate).

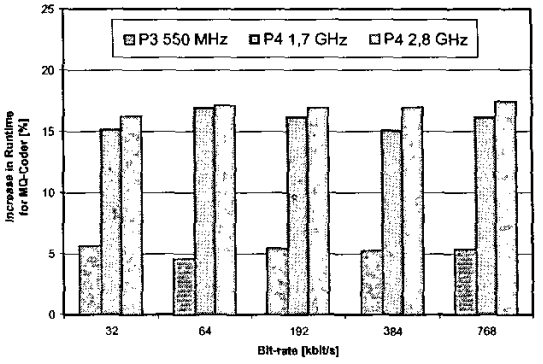

Figure 3: Average increase in runtime in % by using the MQ coder instead of the M coder in a real-time H.264/AVC decoder (measured for the arithmetic decoding engine part only).

The first set of experiments aimed at evaluating the rate-distortion performance of both the original JM50c including the M coder and the MQ coder adapted version. In addition, a conventional binary arithmetic coder using the exact arithmetic in (1) to perform the interval subdivision [4] as well as an alternative configuration of the M coder with $(K, N) = (2, 47)$ have been tested for the test set consisting of 4 sequences in QCIF resolution and 3 sequences in CIF resolution. The graph in Fig. 1 shows the average bit-rate savings for the three different binary coders relative to the M coder configuration of the H.264/AVC standard. As can be seen from the graph, the performance of the standardized M coder and the conventional arithmetic coder using a scaled-count estimator [4],[5] is virtually identical for the tested material. For the MQ coder, however, an average increase in bit-rate between 2–4% has been observed, while the performance of the alternative M coder configuration with the same table size of the MQ coder ($2 \cdot 47$ bytes) results only in approx. 1% loss in coding efficiency relative to the original M coder configuration of H.264/AVC.

A second set of experiments was performed in order to compare the speed of both coding engines in a real-time decoder on different platforms. For this purpose, runtime measurements were made using an optimized software-based H.264 decoder implementation running on a Windows PC. Three different processor platforms have been used for our tests: Pentium 3 (P3) at 550 MHz, Pentium 4 (P4) at 1.7 GHz, and P4 at 2.8 GHz. As test sequences, we used two QCIF sequences coded at 32 and 64 kbit/s, and three CIF sequences coded at 192, 384, and 768 kbit/s.

The graph in Fig. 3 shows that on the average a 4.6–17.5% increase in runtime of the arithmetic decoding engine has been observed for the MQ coder variant of the real-time decoder implementation compared to the original version including the H.264 M coder. For the P4 processor platform, the average MQ coder runtime was increased substantially by 15–17.5% relative to the total runtime of the M coder measured at virtually the same total bit-rate. A consistently smaller increase in runtime was measured for the MQ coder on the P3 platform. One reason for this behavior might be attributed to the more limited cache size of the P3 architecture such that the smaller sized lookup table of the MQ coder might be of some advantage.

4. CONCLUSIONS

A new fast and efficient algorithm of binary arithmetic coding was presented, which offers a great flexibility in the design to serve the different needs of various applications. Information was provided to motivate the specific configuration of the proposed coder as a normative part of the H.264/AVC video standard. Experimental results have shown that our proposed design provides advantages both in terms of compression performance and speed, when compared to the state-of-the-art MQ coder.

REFERENCES

[1] Pennebaker, W.B., Mitchell, J.L., Langdon, G.G., and Arps, R.B., "An Overview of the Basic Principles of the Q-Coder Adaptive Binary Arithmetic Coder", *IBM J. Res. Dev.*, Vol. 32, pp. 717-726, 1988.

[2] Rissanen, J. and Mohiuddin, K.M., "A Multiplication-Free Multialphabet Arithmetic Code", *IEEE Trans. Commun.*, Vol. 37, pp. 93-98, Feb. 1989.

[3] Howard, P.G. and Vitter, J.S., "Practical Implementations of Arithmetic Coding", in *Image and Text Compression*, J. A. Storer (Ed.), Kluwer, 1992, pp. 85-112.

[4] Moffat, A., Neal, R.M., and Witten, I.H., "Arithmetic Coding Revisited", *ACM Trans. on Inf. Sys.*, Vol. 16, No. 3, pp. 256-294, July 1998.

[5] Duttweiler, D.L. and Chamzas, C., "Probability Estimation in Arithmetic and Adaptive-Huffman Entropy Coders", *IEEE Trans. on Image Proc.*, Vol. 4, pp. 237- 246, 1995.

[6] Taubman, D. and Marcellin, M.W., *JPEG2000 Image Compression: Fundamentals, Standards and Practice*, Kluwer Academic Publishers, 2002.

[7] Marpe, D., Schwarz, H., Wiegand, T., "Context-Based Adaptive Binary Arithmetic Coding in the H.264/AVC Video Compression Standard", *IEEE Trans. on Circ. and Systems for Video Technology*, to be published.

[8] Wiegand, T. and Sullivan, G., "Draft Text of Final Draft International Standard (FDIS) of Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC)", JVT-G050, March 2003.

[9] Univ. of British Columbia, http://spmg.ece.ubc.ca/jbig2