FAST RENORMALIZATION FOR H.264/MPEG4-AVC ARITHMETIC CODING

Detlev Marpe, Heiner Kirchhoffer, and Gunnar Marten

Fraunhofer Institute for Telecommunications – Heinrich Hertz Institute, Image Processing Department
Einsteinufer 37, D-10587 Berlin, Germany
phone: +493031002-619, fax: +49303927200, email: [marpe|kirchhoffer|marten]@hhi.fraunhofer.de
web: iphome.hhi.de

ABSTRACT
We propose a fast, standard-compliant realization of the computationally expensive renormalization part of the binary arithmetic coder in H.264/MPEG4-AVC. Our technique allows to replace time-consuming, bitwise-operating input and output procedures as well as bitwise carry-over handing in a conventional implementation with corresponding operations in units of multiple bits. Experimental results demonstrate that the proposed method enables a considerable speed-up of both arithmetic encoding and decoding in the range of 24–53% of their corresponding run time.

1. INTRODUCTION

In the course of the standardization of H.264/MPEG4-AVC [1], a novel design of a family of table-based adaptive binary arithmetic coders has been developed [2]. This so-called M-coder design [3] involves the innovative features of a table-based interval subdivision in conjunction with a fast and accurate table-based probability estimator as well as a fast bypass coding mode. The computationally critical operation of interval subdivision is approximated by using a pre-quantization of the range of possible interval width values induced by renormalization. For each quantized interval width and for each representative probability value, the corresponding product value is pre-calculated and stored with suitable precision into a 2-D lookup table. Probability estimation is performed by employing a finite-state machine with tabulated transition rules. For approximately uniform distributed sub-sources, an optional bypass of the probability estimator is employed, which results in an additional speed-up [3][4].

A member of the M-coder family has been adopted as normative element of the H.264/MPEG4-AVC context-based adaptive binary arithmetic coding (CABAC) scheme [4]. This specific M-coder incarnation in H.264/MPEG4-AVC provides virtually the same coding efficiency as a conventional multiplication- and division-based implementation of binary arithmetic coding at significantly higher throughput rates, corresponding to speed-up factors in the range of 1.5–2.0 [2]. Compared to other well-established low-complexity binary arithmetic coding techniques like that of the Q coder [5] and its derivatives of QM and MQ coder [6], the M coder achieves an increase in throughput of up to 18%, depending on the implementation platform. At the same time, average bit rate savings of 2–4% can be obtained by the M coder relative to the MQ coder, when measured in the native H.264/MPEG4-AVC CABAC environment [3].

Despite these remarkable properties, arithmetic coding in H.264/MPEG4-AVC still poses some severe problems for real-time applications. Due to its sequential nature of processing, the computational requirements for real-time software-based parsing and arithmetic decoding of high-definition video at, e.g., bit rates of 8–12 Mbits/sec, may yet exceed the capabilities of today’s generic CPUs. In the view of these challenges, it is obvious that any progress in substantially reducing the implementation costs of the binary coding engine of H.264/MPEG4-AVC will be extremely beneficial.

One of the major bottlenecks in any arithmetic encoding and decoding process is given by the renormalization procedure. Renormalization in the M coder is required whenever the new interval range \( R \) after interval subdivision does no longer stay within its legal range. Each time a renormalization operation must be carried out one or more bits can be output at the encoder or, equivalently, have to be read by the decoder. This process, as it is currently specified in the standard [1], is performed bit-by-bit, and it is controlled by some conditional branches to check each time if further renormalization loops are required. Both conditional branching and bitwise processing, however, constitute considerable obstacles to a sufficiently high throughput.

As a solution to this problem, we propose a fast renormalization policy for the M coder with the following main characteristics:

- The loop in the renormalization part is completely removed and conditional branching is omitted as far as possible.
- The internal register representing the code interval base \( L \) in the encoder or alternatively, the register for the offset \( V \) in the decoder is implemented with a higher accuracy in order to allow writing/reading of multiple code bits at a time.
- The carry-over handling in the encoder is substantially simplified in a way that the demand for storage and computational resources can be greatly reduced.
- A virtual floating point is maintained for the registers \( L \) and \( V \) to always guarantee the required precision of the corresponding variables relative to the code interval width \( R \).
- All proposed changes as applied to the H.264/MPEG4-AVC version of the M coder are fully standard compliant.
2. BACKGROUND AND PROBLEM STATEMENT

In the following, we first present a brief review of the basic principles of binary arithmetic coding (BAC) with a particular focus on implementation-related aspects. In binary arithmetic coding, it is convenient to discriminate between the two symbols of the binary alphabet not by using their actual symbol values “0” and “1” but rather by referring to their estimated probability values. By distinguishing between the least probable symbol (LPS) and the most probable symbol (MPS) and by keeping track of the symbol value of the MPS (valMPS) as well as the probability (MPS) and by keeping track of the symbol

\[ \text{val} = \text{valMPS} \]

If the probability estimation involves a simple estimator based on scaled cumulative frequency counts of symbols, this operation may even involve an integer division \([7]\). As a consequence, most of the research on fast binary arithmetic coding has been devoted to the problem of employing a suitable low-complexity operation in an approximation of the operation(s) required to perform the interval subdivision. The most prominent representatives of that kind of BAC schemes are given by the PQ, QM and MQ coder as part of JPEG, JBIG, JPEG-LS, and JPEG2000 image coding standards \([5][6]\).

Recently, a new design of a family of multiplication-free binary arithmetic coders has been proposed \([2][3]\). Its main innovative features are given by a table-based interval subdivision coupled with probability estimation based on a finite-state machine (FSM) as well as a fast bypass coding mode. This so-called modulo (M) coder family of BAC schemes offers a parameterizable trade-off between coding efficiency and memory requirements for the underlying lookup tables. Actually, the M-coder design can be considered as a generalization of the Q-coder family\(^1\), since the latter can be derived from a specific M-coder incarnation belonging to the simplest choice of parameter (see below).

Another, more elaborate choice of a member of the M-coder family has been adopted by the ITU-T and ISO/IEC as a normative part of the H.264/MPEG4-AVC video coding standard \([1]\). It offers a good trade-off between complexity (in terms of throughput) and compression performance, as experimentally verified in \([3]\). In the following section, we briefly summarize some basic facts of the M coder.

2.1 Brief review of the M-coder design principles

The basic idea of the low-complexity M-coder approach of interval subdivision is to quantize the admissible domain \(D = [2^{b-2} , 2^{b-1}]\) for the range register \(R\) induced by renormalization into a small number of \(K\) different cells. To further simplify matters, we assume a uniform quantization of \(D\) to be applied, resulting in a set of representative equi-

\[ \text{val} = \text{valMPS} \]

1 This is strictly true only with regard to the way the interval subdivision is approximated.
spaced range values $\mathcal{L} = \{Q_0, Q_1, \ldots, Q_{K-1}\}$, where $K$ is further constrained to be a power of 2, i.e., $K = 2^\kappa$ for a given integer $\kappa \geq 0$. By a suitable discretisation of the range of LPS-related probability values $p_{\text{LPS}} \in (0, 0.5]$, a representative set $\mathcal{P} = \{p_0, p_1, \ldots, p_{M-1}\}$ of probabilities can be constructed together with a set of corresponding transition rules for FSM-based probability estimation. Both discrete sets $\mathcal{P}$ and $\mathcal{L}$ together enable an approximation of the multiplication operation $p_{\text{LPS}} \times R$ for interval subdivision by means of a 2-D table $\text{RTAB}$ that contains all $M \times K$ pre-calculated product values $\{p_m \times Q_k \mid 0 \leq m < M; 0 \leq k < K\}$ in a suitably chosen integer precision. The entries of the $\text{RTAB}$ table can be easily addressed by using the (probability) state index $m$ and the quantization cell index $k$ related to the given value of $R$. Computation of $k$ is easily carried out by a concatenation of a bit-shift and a bit-masking operation applied to $R$, where the latter can be interpreted as a modulo operation using the operand $K = 2^\kappa$, hence the naming of the proposed family of coders:

$$k = (R >> (b - 2 - k)) \& (2^\kappa - 1). \quad (1)$$

Please note that for a specific realization of the M coder, $\kappa$ and $b$ are fixed, and therefore both operands on the right hand side of (1) are given as fixed values. By choosing a value of $\kappa = 0$, the 2-D table $\text{RTAB}$ degenerates to a linear table, for which all possible values of $R$ only one single representative value is used for the approximation of $p_m \times R$. This case is equivalent to the subinterval division operation performed in the Q coder and its corresponding derivatives.

However, for clarity of presentation and without loss of generality, we will restrict ourselves in the following to the specific case of an H.264/MPEG4-AVC compliant M coder corresponding to the choice of $\kappa = 2$ and the specification of a lookup table $\text{RTAB}$ with $64 \times 4$ entries [1]. As a further simplification, we will neglect the table lookup operations required to adapt the probability state $m$ during each encoding/decoding cycle. For more details, especially on the latter aspect, please refer to [3][4].

2.2 Discussion of renormalization

In terms of implementation costs, the renormalization part of the M coder still suffers from bit-by-bit input/output and - as far as the encoder side concerns - also from bitwise carry-over handling. The related computationally critical parts in an encoder implementation can be mainly attributed to the bitwise operating renormalization loop and the conditional branching inside this loop as shown in Fig. 1.

Although from a decoder perspective, the problem appears to be slightly alleviated when comparing the renormalization parts of Fig. 1 and Fig. 2, there is still a considerable computational overhead involved in a conventional M-decoder implementation due to its sequential reading of bits from the bitstream (as exemplified in line no. 11 of Fig. 2).

The following section presents an alternative but still standard-compliant realization of renormalization by enabling the processing of multiple bits at a time both for the output at the encoder and the input at the decoder.

```c
// interval subdivision
1: $R_{\text{LPS}} = \text{RTAB}[m][R \gg 6] & 3$
2: $R_{\text{LPS}} = R - R_{\text{LPS}}$
3: if ($V < R_{\text{LPS}}$)
4: $R = R_{\text{LPS}}$
5: else
6: $V = V - R_{\text{LPS}}$
7: $R = R_{\text{LPS}}$

// renormalization
8: while ($R < 2^9$)
9: $R = R << 1$
10: $V = V << 1$
11: $V = V | \text{read_one_bit}()$
```

Fig. 2 – Implementation of an H.264/MPEG4-AVC compliant M decoder w/o probability estimation (using a fixed probability state $m$).

3. FAST STANDARD-COMPLIANT RENORMALIZATION

3.1 Determination of renormalization cycles

The first natural step toward a simplification of renormalization consists in unrolling the while loop (line no. 8 of both Fig. 1 and Fig. 2). It is quite obvious that for avoiding multiple checks of the while condition, it is sufficient to determine in advance the bit index of the most significant bit (MSB) in the $R$ register relative to the loop guard with its MSB placed at bit index equal to 8 (for the specific M coder under consideration). Since the value of $R$ is doubled or left-shifted for each loop cycle, the numerical difference between 8 and the current bit index of the MSB of $R$ is equal to the number of cycles the renormalization loop has to be executed.

Many hardware architectures allow to determine the MSB bit index within a single instruction like, e.g., the Bit Scan Reverse (BSR) instruction on Intel’s x86 architecture [8]. However, in cases where the implementation of the M coder has to be more generic or platform-independent, the use of such low-level machine dependent instructions may be prohibited. For those use cases or simply for cases where no specific instructions for MSB index detection are available, we propose an alternative way of determining the number of renormalization cycles.

To this end, we first discriminate between the MPS and LPS case. In case of encoding/decoding an MPS event, the value of $R_{\text{MPS}} = R - R_{\text{LPS}}$ can be bounded from below as follows. Let us assume that according to equation (1), a specific value of $k$ with $0 \leq k < 4$ is derived from $R$. Then, the estimation $R \geq (4+k) \times 2^8$ holds and from the specification of the $\text{RTAB}$ table in [1], we can deduce the following upper bounds for $R_{\text{LPS}}$, depending on the value of $k$:

$$R_{\text{LPS}} = \text{RTAB}[m][k] \leq \begin{cases} 128, & \text{if } k = 0 \\ 176 + (k - 1) \times 2^3, & \text{else} \end{cases}, \quad (2)$$

for all $m$ with $0 \leq m < 64$. Combining both estimations, we can conclude that $R_{\text{MPS}} \geq 128$ always holds and therefore, at most one renormalization cycle is required for the MPS. This is equivalent to what one would expect from an exact implementation of subinterval division as outlined in Sec. 2.
1: \( R_{\text{R}} = \text{RTAB}[m] \) \( (R >> 6) \& 3 \)
2: \( R_{\text{R}} = R - R_{\text{R}} \)
3: if \( (V < (R_{\text{R}} << \text{BitsLeft})) \)
4: \( R = R_{\text{R}} \)
5: \( \text{Rnorm} = (R_{\text{R}} >> 8) \) XOR 1
6: else
7: \( V = V - (R_{\text{R}} << \text{BitsLeft}) \)
8: \( R = R_{\text{R}} \)
9: \( \text{Rnorm} = \text{RnormTAB}[R_{\text{R}} >> 3] \)
10: \( R = R << \text{Rnorm} \)
11: \( \text{BitsLeft} = \text{BitsLeft} - \text{Rnorm} \)
12: if \( (\text{BitsLeft} <= 0) \)
13: \( V = (V << \text{M_BITS}) \mid \text{read_bits(M_BITS)} \)
14: \( \text{BitsLeft} = \text{BitsLeft} + \text{M_BITS} \)

Fig. 3 – Proposed fast renormalization in an H.264/MPEG4-AVC compliant M decoder (w/o probability estimation).

because from the definition we have \( p_{\text{MPS}} \geq 0.5 \geq p_{\text{LPS}} \). Thus, for the MPS, a simple bit test is sufficient to compute the number of renormalization cycles, denoted by \( \text{Rnorm} = (R_{\text{MPS}} >> 8) \) XOR 1, where XOR denotes the logical exclusive-or operation.

In the LPS case, we can directly deduce \( \text{Rnorm} \) from the tabulate \( R_{\text{R}} \) values of \( \text{RTAB} \). A straightforward method, for instance, would be to put the corresponding \( \text{Rnorm} \) values in a complementary 2D-table with 64 \( \times \) 4 entries. However, by observing that the entries of \( \text{RTAB} \) imply a strict lower bound for \( R_{\text{R}} \), a much smaller and hence more practical lookup table can be derived as follows.

First, from the definition of the underlying FSM of the M coder [1][3][4], it is clear that the values of \( p_m \in \mathcal{P} \) are given in decreasing order with increasing value of the probability state \( m \). This, in turn, implies that the maximum value of \( \text{RTAB}[m][k] \) is given for the maximum value of \( m \). The probability state with \( m = 63 \), however, corresponds to an autonomous, non-adaptive state within the H.264/MPEG4-AVC-based M-coder realization, and it is only used for encoding/decoding of terminating syntax elements, for which often a separate encoding/decoding routine is utilized [1][4].

Since \( \text{RTAB}[63][k] = 2 \) for all values of \( k \), the corresponding \( \text{Rnorm} \) value can be determined to be equal to 7. For all regular states of the FSM corresponding to \( m < 63 \), we have \( R_{\text{R}} = \text{RTAB}[m][k] \geq 6 \). Based on this lower bound for all states with \( m < 63 \), we can aggregate the \( \text{Rnorm} \) values that correspond to \( R_{\text{R}} \) values strictly less than 8, because for those values exactly 6 renormalization cycles have to be performed. Consequently, we can discard the 3 least significant bits of \( R_{\text{R}} \) and use the remaining 5 MSBs for indexing a table \( \text{RnormTAB} \) which is constructed to indicate the unique number of renormalization cycles for each value of \( R_{\text{R}} \) (with the exception of those related to \( m = 63 \)):

\[
\text{Rnorm} = \text{RnormTAB}[R_{\text{R}} >> 3]
\]

Note that \( \text{RnormTAB} \) is a table that requires not more than 31 entries, each with a precision of 3 bits only. This follows from the upper bound on the \( \text{Rnorm} \) value (equal to 6 for

---

2 Note, however, that due to the approximations involved, it is not always guaranteed that \( R_{\text{MPS}} \geq R_{\text{LPS}} \) for the M coder.
stream enables a significantly simplified carry-over processing. Fig. 4 shows an encoder implementation equipped with the proposed multiple-bit output and carry-over handling.

4. EXPERIMENTAL RESULTS

For an experimental evaluation of our proposed fast renormalization scheme, we first implemented the decoder related changes into our own run-time optimized H.264/MPEG4-AVC High profile (HP) decoder. In addition to what has been discussed in the previous section, we also adapted the initialization, the termination as well as the bypass part of the M decoder according to the modified renormalization strategy. For that implementation, a value of 16 was chosen for $M\text{BITS}$. We generated H.264/MPEG4-AVC HP-conforming bitstreams for four 1080p test sequences (each with 50 frames) by using intra-only coding with fixed QPs of 20 and 24. These settings were chosen because for a typical HDTV broadcast scenario, the generated bit-rates can be regarded as a kind of upper bound for HP@L4 [1].

For the purpose of performing reliable run-time measurements, we disabled the decoding process and measured the parsing process only. The run-time measurements for parsing the test bitstreams were performed by using our H.264/MPEG4-AVC HP decoder, comparing a conventional M decoder implementation with our proposed fast renormalization scheme. The corresponding experiments were performed on a Pentium 4, 3.6 GHz machine with Linux OS, where the code was generated using gcc, version 4.03.

As a result of those experiments, we obtained an average reduction in measured run time for the M decoder part in the range of 23.5–26.9%. Note that the proportion of the actual arithmetic decoding process relative to the whole parsing process was about 35%.

In another set of experiments, we compared run time for two different encoder versions – one conventional implementation and another version using the proposed fast renormalization. These experiments were carried out by using a JPEG still image coding implementation [9], where the corresponding QM coder was replaced by our two versions of the M coder. The reason for using JPEG instead of H.264/MPEG4-AVC for these experiments is given by the fact that in a typical H.264/MPEG4-AVC encoder, the proportion of run time of the arithmetic encoding part is usually too small to deliver statistically reliable results. As an outcome of our JPEG-based encoding experiments, we obtained average run-time improvements for the M encoder of 47.3–52.7%, where parts of these relatively large gains can be attributed to the largely improved carry-over handling.

5. CONCLUSIONS

We have introduced a fast, standard-compliant renormalization method for the binary arithmetic encoder and decoder in H.264/MPEG4-AVC. By replacing the conventionally bit-wise performed operations with byte-wise or word-wise processing, a considerably increased throughput can be achieved. We presented experimental results for demonstrating the benefits of the novel renormalization technique, especially for the purpose of software-based decoding of HD video.

REFERENCES


