Rate and Decoding Power Constrained Video Coding Scheme for Mobile Multimedia Players

Ligang Lu and Vadim Sheinin
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598

ABSTRACT

Power dissipation, data rate, and processing time are crucial constraints for embedded multimedia devices. We present a video coding solution given the constraints of bit rate and decoding power dissipation. Coupled with a small dedicated video internal memory at a decoder, the coding scheme is designed for the best operational rate-distortion and decoding power cost trade off. It can substantially decrease the data traffic to the external memory at decoder. A decrease in data traffic to the external memory at decoder will result in faster real-time processing and power savings. The encoder, given the prior knowledge of the decoder's dedicated video internal memory management scheme, optimally regulates its choice of motion compensated predictors to reduce the decoder's external memory accesses. This video coding scheme can be used in any standard or proprietary encoder to generate a compliant output stream decodable by a standard general purpose processor-based or dedicated hardware-based decoder with power restriction. Simulation results show that with a relatively small dedicated internal memory, our scheme may reduce the power dissipation significantly while keeping the comparable picture quality.

INTRODUCTION

Embedded mobile devices, such as smart phones, Personal Digital Assistants (PDAs), etc. are increasingly becoming multimedia players. Video decoding is often an essential part of a multimedia application. Standard video coding algorithms, such as MPEG[1,2], H.26x use motion compensated temporal prediction techniques. One common requirement for decoders to use these schemes is a large external memory to store reconstructed video frames as motion compensated prediction references in decoding. Whether the decoder implementation is based on a general purpose processor or a dedicated hardware solution, the cost of accessing external memory is expensive both in terms of real-time and system power consumption. These factors are very important for embedded mobile multimedia devices. The complexity of multimedia algorithms increases from generation to generation. For example, the video decoding algorithm H.264 (also known as MPEG-4 Part-10) is several times more complex than the "initial" MPEG-4 Part-2. Although video quality is superior by using H.264, there are tough challenges in implementing the algorithm on existing platforms, regardless it is a general processor or dedicated hardware decoder. Although higher processing frequency can increase real-time performance, it also results in higher power dissipation and higher silicon cost. Embedded multimedia devices such as smart phones, PDAs, etc. are very cost sensitive and power restrictive. Therefore other means for performance improvement are needed. In this paper we extend our previous results[4] further describe a rate and decoding power constrained video coding scheme coupled with a dedicated video internal memory(DVIM) at the decoder. The scheme can substantially reduce the external memory access in decoding process hence to achieve significant savings in power dissipation and run-time for embedded multimedia devices.

2. VIDEO DECODER WITH DADICATED VIDEO INTERNAL MEMORY AND ITS REGULATION

Video compression algorithms commonly use the temporal prediction scheme to exploit the temporal redundancy due to the fact that the current frame of video can often be largely predicted from previously decoded frames, thus only the prediction error needs to be encoded and transmitted to the decoder for reconstructing the current frame. The frames used to form such prediction are called the reference frames. In the natural display order, reference frames can either temporally precede or succeed the frame being decoded. Furthermore, most video coding standards use a block-based prediction structure, wherein the prediction for a block in a frame is formed from a corresponding block of data in a reference frame specified by one or more motion vectors. In a typical video decoder, reference frames are too large to be wholly accommodated in the on-chip internal memory. So forming the motion compensated reconstruction involves:

- Fetching the relevant reference blocks of data specified by the motion vectors from the off-chip external memory to the internal memory;
- Performing motion compensated reconstruction by adding the decoded prediction error (texture) to the reference block;
- Writing the reconstructed block to the external memory

For example, in the MPEG-2 video standard, each forward-predicted 16x16 block encoded in the frame mode needs at least a 16x16 block from the reference frame to form a reconstruction. On average, some parts of the reference frame are fetched multiple times in the process of prediction while some part is not used at all.

In a typical video decoder, the motion vectors and the texture or the prediction error are decoded first. The decoded motion vectors specify the reference block in the previously reconstructed reference frame that was used to make the temporal prediction in encoding the current block. The reconstructed reference frames is generally stored in external memory because it is unwise to keep this large memory inside a reasonable chip. According to the motion vectors,

0-7803-8554-3/04/$20.00 ©2004 IEEE.

the reference data is fetched from the external memory into the on-chip internal memory for motion compensated reconstruction, where the decoded prediction error block and the reference block are added with proper clipping. Finally the reconstructed block is stored in the external memory. Such video decoder requires fetching at least one block from the reference frame in the external memory for decoding every predictive-coded block. Depending on the implementation, it often needs to fetch more than one block of reference data if the reference block is not aligned with the block boundary. Fetching data from the external memory not only is much slower than from the internal memory but also is more costly in power consumption. However, both real-time performance and power dissipation are the most crucial factors for embedded mobile multimedia devices.

2.1 Video Decoder with Dedicated Video Internal Memory

To reduce the power dissipation and increase the real-time processing performance in video decoding, we can take the advantage of the fact that some of the reference block may be entirely or partially used more than one time in decoding different blocks in the frame. If we maintain a dedicated video internal memory (DVIM) at the video decoder to keep some used reference blocks for possible reusing in decoding the remaining blocks of the current frame, the number of fetching reference data from the external memory can be reduced and consequently both the decoding run-time and power consumption can be reduced. Fig. 1 shows such a video decoder with a DVIM for this purpose. In Fig. 1, the general decoding process is largely the same as a typical decoder. The difference is that after decoding a block, the decoder will store the used reference data into DVIM if it has not been stored yet and when decoding the next block, the decoder looks for the reference block specified by the motion vectors in DVIM first, if the reference data is in DVIM, the decoder will fetch it from DVIM instead of from the external memory. The advantages to use the DVIM cache to hold some previously used reference data are that the access to it from either a general purpose processor-based or dedicated hardware-based video decoder is very fast and consumes much less power. It should be noted that DVIM is not a general CPU cache in the case of a general processor-based decoder. One cannot use a general cache because video decoding is not necessary the only task running on the CPU; there are others tasks, such as audio decoding, scheduling, etc. DVIM here is a dedicated video-only fast memory.

2.1 Management Scheme of DVIM

Many methods can be used to regulate the operation of the DVIM. One simple method is to use a look-up table. The table maintains the block index which implies its location or address in the reference frame, a 1-bit status variable indicating whether the reference block is currently in the DVIM, and the address of the given block in DVIM.

At the beginning of decoding a frame, DVIM is empty. After the motion vectors specify one or more reference blocks needed for motion compensation, the decoding routine will first check the look-up table to see if any of the reference blocks is in DVIM. If a needed reference block is already in DVIM, it will be taken from DVIM for reusing. Otherwise, the reference block will be fetched from the external memory and the new reference block will be stored into DVIM after use and the look-up table will be updated accordingly. When DVIM is full and there are new reference blocks to be stored, one can use various methods for DVIM and look-up table management. For example, one simple method is based on the principle of first-in-first-out approach; wherein the first stored reference block in DVIM will be replaced first. A more sophisticated method is to replace the reference block whose position is above the motion estimation search range.

3.VIDEO CODING SCHEME WITH RATE AND DECODING POWER CONSTRAINTS

Now we describe a video coding scheme which, coupled with the decoder's DVIM, can significantly reduce the video decoder's external memory access numbers and thus reduce its power dissipation and improve its real-time performance in video decoding. More specifically, knowing the decoder's DVIM model, i.e., its size and operation managing scheme, the encoder can emulate the exact operation of DVIM and control the content of DVIM. Therefore the encoder can, to a certain extend, control the number of the external memory accesses in decoding; in another word, in the encoding process, the encoder can trade off the distortion with both the bit rate and the decoder power constraints.

3.1 Problem Formulation

To facilitate the analysis, but without losing generality, let us consider to encode a video frame using MPEG-4 forward prediction frame coding mode, i.e., a P frame. Giving a frame bit target usage budget R and a frame decoding power dissipation cost budget C, the encoder tries to minimize the coding distortion D subject to the constraints of R and C:

$$Min\, D \quad s.t. R, \quad C. \tag{1}$$

Denote R_i the rate and C_i the power cost associated with fetching the reference data from the external memory for the i-th block decoding. Assuming the additive property of the distortion, rate, and power cost over the blocks in the frame, we have

$$Min \sum_{i=1}^{N} D_i \quad s.t. \sum_{i=1}^{N} R_i \le Rand \quad \sum_{i=1}^{N} C_i \le C \tag{2}$$

where N is the number of blocks coded by forward prediction in the frame. Apply the well-known Lagrangian minimization technique, the objective functional is

$$J = \sum_{i=1}^{N} D_i + \lambda \sum_{i=1}^{N} R_i + \beta \sum_{i=1}^{N} C_i \tag{3}$$

where λ, β are the Lagrangian Multipliers that weight the importance of the rate and decoding power cost. For example a large β means that the decoder has a very high power dissipation constraint while $\beta = 0$ indicates that the power cost at the decoder is not a concern. To solve the Lagrangian Functional in (3) directly and jointly will impose a heavy computation burden for real-time economic implementations. Here we propose a simplified approach for practical real-time implementations for MPEG and H.26x applications. Examining (3) we can see that the last term in the functional imposes a limitation on the selection of the motion vectors

and which in turn will affect the motion compensated prediction errors or the residues. On the other hand, the smaller the prediction errors, the better the rate-distortion results, i.e., the smaller the sum of the first two terms in (3). Suppose that, giving the motion vectors, the encoder has a rate-distortion optimization algorithm that will minimize distortion *D* subject to the bit target *R*, which is generally true for a good encoder, then we may solve the problem in (3) in two steps:

Step 1. Minimize the motion compensated prediction errors *E* subject to *C*. That is

$$Min\sum_{i=1}^{N} E_{mv_i} \qquad s.t \qquad \sum_{i=1}^{N} C_i \le C \tag{4}$$

where E_{mv_i} is the motion compensated prediction error using the motion vector mv_i. The corresponding Lagrangian functional is

$$J_1 = \sum_{i=1}^{N} E_{mv_i} + \beta \sum_{i=1}^{N} C_i . \tag{5}$$

Step 2. Given MV_i or E_{mv_i}, minimize the quantization distortion $D = \sum_{i=1}^{N} D_{mv_i}$ subject to *R*. That is

$$J_2 = \sum_{i=1}^{N} D_{mv_i} + \lambda \sum_{i=1}^{N} R_i . \tag{6}$$

This two-step approach can be much more easily implemented since the rate-distortion optimization algorithm of the encoder is assumed to achieve Step 2; we only need to modify the motion estimation algorithm to achieve Step 1, which is much simpler than directly solving (3) jointly.

3.2 Modified Motion Estimation Scheme for Decoder Power Reduction

Using the two-step approach formulated above, we can readily modify the motion estimation process in the encoder to achieve (5). Fig. 2 describes the modified motion estimation process. The motion estimation has the following major function units: the motion estimator, the DVIM emulator, the reference frame stored in the external memory, the motion vector selector, and the prediction error and decoder power cost controller. The Motion estimator uses a searching algorithm to find the best match between the input block and a block in the reference frame. The DVIM emulator performs the same operation as the DVIM in the decoder. The DVIM emulator stores the reference blocks that were used in coding the previous blocks in the current frame. When the DVIM is full, a previous stored reference block will be replaced by the last used reference block if it has not been stored. The replacement rule is the same as that at the decoder as described in Section 2. The exact motion estimation algorithm is not standardized as long as the output bit stream complies with the underlining standard. Encoders usually perform motion estimation on the reconstructed reference frame to avoid the drift problem between the encoder and decoder. The motion vector selector determines how to encode the input block, i.e., whether to encode the block in intra or inter mode and to use which and how many motion vectors. The prediction error and decoder power cost controller trades off the prediction error versus the reference block reuse or the decoding power cost.

To encode a block in the current frame, the motion estimator first determines the motion vectors between the given block and the best match in the reference frames as the motion estimation without the modification. Next the motion estimator also searches the DVIM emulator to find the best match candidate among the stored previously used reference blocks. Then prediction error and decoder power cost controller compares the two motion compensated prediction errors to see if

$$E_{mv_i^*} + \beta C_i < E_{mv_{DVIM}^*} \tag{7}$$

If (7) holds, then the motion vector mv_i^* will be kept; otherwise mv_{DVIM}^* will be kept, which will reuse the reference data in DVIM. The encoder then proceed to Step 2 to determine the coding mode and quantization for the block according to its rate-distortion optimization algorithm to minimize (6).

4. Simulation Results

The above video coding scheme has been simulated using a MPEG-4 Simple Profile software encoder and tested on CIF (Common Intermediate Format) video sources with 352x288 resolution. The sequences were coded with a fixed quant step 8. Since the actual external memory fetching cost depends on the particular hardware platform, for convenience but without losing the validation value of the simulation, we assume that C_i is a constant *C* for all *i*. Furthermore, the Lagrangian Multiplier β is a variable, which sweeps from 0 to ∞ to find all the operational optimal points. So the exact value of *C* is not necessary to know in the simulation. Now (7) becomes

$$E_{mv_i^*} + \beta C < E_{mv_{DVIM}^*} \tag{8}$$

where βC is a real number.

Fig. 3 shows the external access savings achieved with various sizes of DVIM. In the simulation, we assumed that the decoder external memory fetch size is 8x8 bytes. In MPEG-4 Simple Profile P-frame coding, a 16x16 macroblock can be coded either as a 16x16 macroblock using 1 motion vector or coded as 4 8x8 block with 4 motion vectors. If a 16x16 reference macroblock is aligned with the block boundary, it is count as 4 8x8 reference block fetches; if it is only horizontally or vertically aligned, it will need 6 8x8 reference block fetches; otherwise it will need 9 8x8 reference block fetches. Likewise, if an 8x8 reference block is not aligned, it will need 4 8x8 reference block fetches; if it is only horizontal or vertical aligned, it will need to fetches 2 8x8 reference blocks. The motion estimation algorithm is based on the Diamond Search Algorithm MVFAST[3]. The results have shown that with a relatively small size of DVIM at the decoder, significant savings in the numbers of the external memory access in video decoding can be achieved. Since accessing internal memory is faster and incurs less

power dissipation than accessing the external memory, these savings in external memory access means a substantial reduction in decoding power dissipation and real processing time, which are the most crucial factors for embedded multimedia devices. Further more as shown in Table 1 the video quality in our scheme is comparable with the normal encoding without DVIM. In Coast Guard case, the full search on DVIM produced better motion vectors then the MVFAST within the search window in the reference frame thus results in better PSNR numbers.

5. Summary and Conclusions

We have described a video coding scheme that can trade the operational rate-distortion performance with the decoding power dissipation and processing time. Utilizing a DVIM at the decoder to store the previously used reference data, the decoder may reuse the reference data in DVIM instead of fetching it from the external memory. Since access to the external memory is slower and incurs more power dissipation than access the internal memory, the reduction in external memory access will result in system power savings and real-time processing performance improvement. Knowing the decoder's DVIM and its managing rule, the encoder can effectively regulates the selection of the motion vectors to achieve the best trade-off between the external memory access (i.e. the decoding power cost) and motion compensated prediction error (i.e., the rate and distortion). The simulation results have shown that with a relatively small amount of DVIM, our scheme may reduce the traffic between CPU and the external memory significantly while keeping the comparable picture quality. The proposed video coding scheme is applicable to both P and B pictures and both luminance and chrominance video data. It can be used in any standard and proprietary video compression using temporal prediction, such as MPEG, H.26x, etc.

REFERENCES

1. ISO/IEC 13818-2 Generic Coding of Motion Pictures and Associated Audio: Video 1995.
2. ISO/IEC 14496-2 Generic Coding of Audio-Visual Objects-Part 2 Visual, 2000.
3. ISO/IEC JTC1/SC29/WG11 N4554 Optimised reference software for coding of audio-visual objects, Dec. 2001.
4. L. Lu and V. Sheinin, "Video Coding for Decoding Power Constrained Embedded Devices," Proceedings of Visual Communications and Image Processing , San Jose, Jan. 2004

DVIM Size	0	60	75	90	120
Cost Guard	32.08	32.66	32.57	32.71	32.36
News	36.26	36.12	36.10	35.90	35.92

Table 1. PSNR (dB) Comparison with Various DVIM Sizes

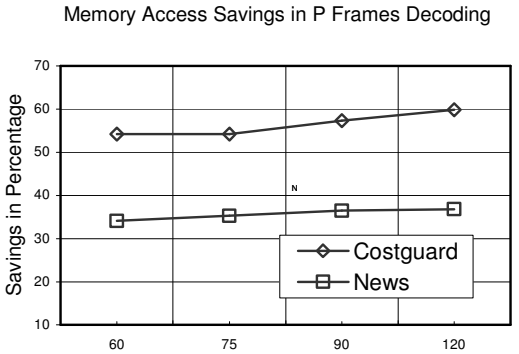

Fig. 3 DVIM Size in 8x8 Blocks

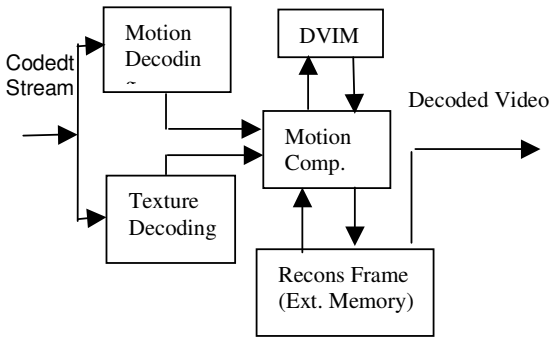

Fig. 1. A Video Decoder with Dedicated Video Internal Memory(DVIM) for Storing Used Reference Blocks.

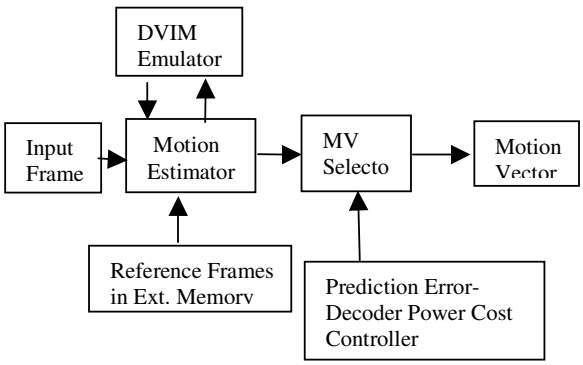

Fig. 2. Motion Estimation with Decoding Power Constraint