RC23071 (W0401-097) January 20, 2004
Electrical Engineering

IBM Research Report

Video Coding for Decoding Power Constrained Embedded Devices

Ligang Lu, Vadim Sheinin
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Video Coding for Decoding Power Constrained Embedded Devices

Ligang Lu and Vadim Sheinin

lul@us.ibm.com

IBM T. J. Watson Research Center

Yorktown Heights, NY 10598

ABSTRACT

Low power dissipation and fast processing time are crucial requirements for embedded multimedia devices. This paper presents a technique in video coding to decrease the power consumption at a standard video decoder. Coupled with a small dedicated video internal memory cache on a decoder, the technique can substantially decrease the amount of data traffic to the external memory at the decoder. A decrease in data traffic to the external memory at decoder will result in multiple benefits: faster real-time processing and power savings. The encoder, given prior knowledge of the decoder's dedicated video internal memory cache management scheme, regulates its choice of motion compensated predictors to reduce the decoder's external memory accesses. This technique can be used in any standard or proprietary encoder scheme to generate a compliant output bit stream decodable by standard CPU-based and dedicated hardware-based decoders for power savings with the best quality-power cost trade off. Our simulation results show that with a relatively small amount of dedicated video internal memory cache, the technique may decrease the traffic between CPU and external memory over 50%.

Keywords: Video Coding, Low Power, Embedded Systems

1. INTRODUCTION

Embedded mobile devices, such as cell phones, Personal Digital Assistants (PDAs), etc. are becoming multimedia-processing systems. Video decoding is often an essential part of a multimedia solution. Standard video compression algorithms, such as MPEG-2[1], MPEG-4[2], H.26x use motion compensated temporal prediction schemes. One common requirement for the decoders to use these schemes is a large external memory for storing video frames at different levels of decoding. Actual platforms to run various decoding algorithms vary from standard general microprocessors to dedicated specific hardware as well as various combinations of the two. Whether the decompression scheme implementation is based on a general 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 characteristics are very important for modern communication systems. The complexity of multimedia algorithms increases from generation to generation. For example, the newest 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 by far superior using H.264, there are very serious challenges in implementing this algorithm on existing platforms, regardless it is a general processor or dedicated hardware. There is not enough real-time performance to address growing demand for resolution, frame rates and lower bandwidth. In order to increase the real-time performance of such platforms, the processing frequency can be increased but it also brings higher power dissipation and higher silicon cost. Many of such multimedia platforms such as smart phones, PDAs, etc. are very cost sensitive; hence other ways for performance improvement should be explored. Additionally, power dissipation is very crucial for all mobile devices, because the battery life is limited and it would have no commercial value for a multimedia enabled PDA that can only play back a multimedia clip for a few minutes. The minimum requirement is usually a few hours.

Existing schemes that use various cache strategies for decoder to improve its performance would not be applicable for the multimedia decoder/player cases; the player includes video decoder, audio decoder, parsing or network adaptation layer as well as synchronization between video and audio. General CPU cache is used for both audio and video data and hence if multiple tasks are running simultaneously, the data for one application in the cache may be contaminated by data from another application.

In this paper we will describe a dedicated video internal memory (DVIM) cache scheme at the decoder coupled with a motion compensated video coding technique at the encoder that will substantially reduce the external memory access at the decoder to achieve significant savings in power consumption and run-time for embedded multimedia devices such as smart phones, PDAs, portable DVD players, etc.

2. THE VIDEO DECODER WITH DADICATED VIDEO INTERNAL MEMORY AND ITS MANAGING SCHEME

Video compression algorithms are commonly using temporal prediction scheme to exploit the temporal redundancy. This is based on the fact that the current frame of video data 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 each block of video data is formed from a corresponding block of data in a reference frame specified by one or more motion vectors. As depicted in Fig. 1, in a typical video decoder system, reference frames are too large to be wholly accommodated in the primary (typically on-chip, internal) memory. So, the process of forming the motion compensated reconstruction block involves:

- Fetching the relevant blocks of data specified by the motion vectors from the secondary (typically off-chip, external) reference frame memory to the primary internal memory;

- Performing motion compensated reconstruction operations by adding the decoded prediction error (texture) to the data of the reference block; and

- Writing the reconstructed block back to a secondary external memory.

For example, in the case of the MPEG-2 compression standard, each forward-predicted 16x16 block encoded in the frame mode needs at least a 16x16 block from the reference frame memory to form a suitable reconstruction. In average, some part of the reference frames is used multiple times in the process of prediction while some part is not used at all.

In the typical video decoder as described in Fig. 1, after the demultiplexing, the incoming bitstream is fed into the video decoder, where 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 require a large amount of memory to store. This storage is generally implemented by means of external memory. It is unwise to keep this large memory inside a reasonable chip. According to the decoded motion vectors, needed 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. Again, both real-time processing speed and power dissipation are two of the most crucial factors for embedded 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 by different blocks in the frame being decoded. 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. 2 shows such a video decoder with a DVIM cache for this purpose.

As shown in Fig. 2, the general decoding process is quite similar as the decoder in Fig.1. 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 which is specified by the motion vectors as shown in Fig. 3a 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 this internal, i.e., "very close" DVIM cache to hold some previously used reference data are that the access to it from either a CPU-based or dedicated hardware-based video decoder is very fast and consumes less power. It should be stressed again that DVIM is not a general CPU cache in the case of a CPU-based system. 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, parsing etc. DVIM here is a dedicated video-only fast memory.

2.1 Management Scheme of DVIM

Many methods can be used to manage or regulate the operation of the DVIM. One simple method is to use a look-up table as shown in Fig.3b. First column of the table is the block index which implies its location or address in the reference frame; the second column is a 1-bit status variable indicating whether the reference block is currently in the DVIM; the third column specifies 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 decoder routine will first check the look-up table of Fig.3b to see if the reference blocks are 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 new reference blocks need 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 (row index) is above the motion estimation search range.

3. VIDEO CODING SCHEME FOR DECODING POWER REDUCTION

In this section we present 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 try to, to a certain extend, minimize the distortion given the bit rate and the decoder power limitation.

3.1 Problem Formulation

To facilitate the analysis, let us consider to encode a video frame using MPEG-4 forward prediction frame coding mode, i.e., a P frame. Denote C_i the cost associated with the *i*-th fetching of an *n*-by-*n* block from the external memory at the decoder. Giving a frame bit usage target *R* and a frame power dissipation cost budget *C*, the encoder tries to minimize the coding distortion *D* subject to the constraints of *R* and *C*:

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

Assuming the additive property of the distortion over the blocks in the frame, we have

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

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 decoder power cost. For example a large β means that the decoder has a very high power dissipation constraint while $\beta = 0$ means that the power cost at the decoder is trivial or 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 in the previous section, we can modify the motion estimation process in the encoder to achieve (5). Fig. 4 describes the modified the encoder motion estimation process. The motion estimation involves 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 no been stored. The replacement can follow to the rules described in Section 2.

The exact motion estimation algorithm is not standardized and the user can use any algorithm as long as the output bit stream complies with the underlining standard. The main feature of motion estimation algorithm is to effectively find the "best" match between the block being coded and a block in a reference frame stored in the external memory. Encoders usually perform motion estimation on the reconstructed reference frame since that is what the decoder has so as to avoid the drift problem between an encoder and a decoder. The external memory holds the reconstructed reference frames for motion estimation and motion compensation purpose. The motion vector selector determines the choice on how to encode the input block, i.e., whether to encode the block in intra mode (without prediction) or inter mode (with motion compensated prediction) 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 input 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 for decoder power reduction has been simulated using a MPEG-4 Simple Profile software encoder and tested on several CIF (Common Intermediate Format) video sources with a resolution of 352x288. Since the actual external memory fetching cost depends on the particular hardware, for convenience and without losing the validation value of the simulation, we assume that C_i is a constant *C* for *i*. Furthermore, the Lagrangian Multiplier β is a variable, which sweeps from 0 to ∞ to find all the theoretical optimal solutions. 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.

CIF Sequence	Coast Guard	News	Mobile	Tempete
External Memory Access Savings %	54.49	34.10	50.26	41.71
PSNR (dB) w/DVIM	32.66	36.12	31.01	31.25
PSNR (dB) w/o DVIM	32.08	36.26	30.80	31.56

Table 1. Simulation Results in P-frames decoding with DVIM size = 60 8x8 blocks, Q=8.

Table 1 shows the some preliminary results on 4 CIF testing sequences: Coast Guard, News, Mobile Calendar, and Tempete. The DVIM size is 60 of 8x8 blocks. In the simulation, we assumed that the decoder external memory fetch size is also 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 in P-frame decoding have shown that with a relatively small size of DVIM at the decoder, 34~54% savings in access numbers to the external memory in video decoding have been 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 processing real time which are the most crucial factors for embedded multimedia devices.

5. Summary and Conclusions

Embedded multimedia devices have crucial constraints on power dissipation and processing time. We have described a video coding scheme that can reduce the power dissipation and processing time in video decoding. Utilizing a DVIM at the decoder to store the previously used reference data, the decoder may reuse the stored 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 increase the real-time processing performance. Knowing the decoder's DVIM and its managing rule, the encoder can emulate the DVIM's operation and consciously regulates the selection of the motion vectors to effectively reduce the external memory access at the decoder. The simulation results have shown that with a relatively small amount of DVIM, our scheme may decrease the traffic between CPU and the external memory by over 50%.

It should be pointed out that the proposed video coding scheme is applicable to both P- and B-pictures and for both luminance and chrominance data of a video frame. Also it can be used in any standard and proprietary video compression using temporal prediction, such as MPEG H.26x, etc. for power savings and real-time performance improvement at both general CPU based or dedicated hardware based decoders.

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.

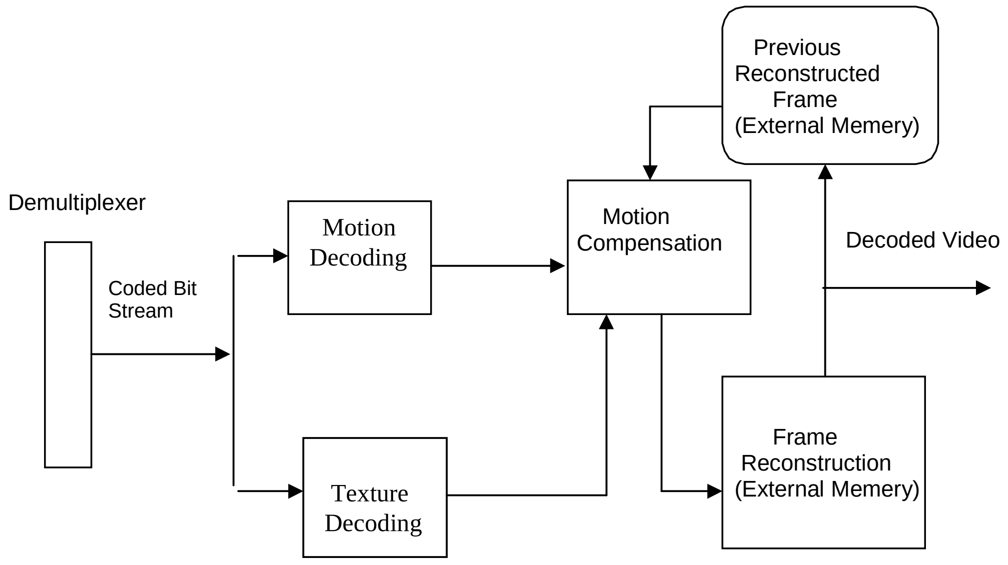

Fig.1 A Typical Video Decoder

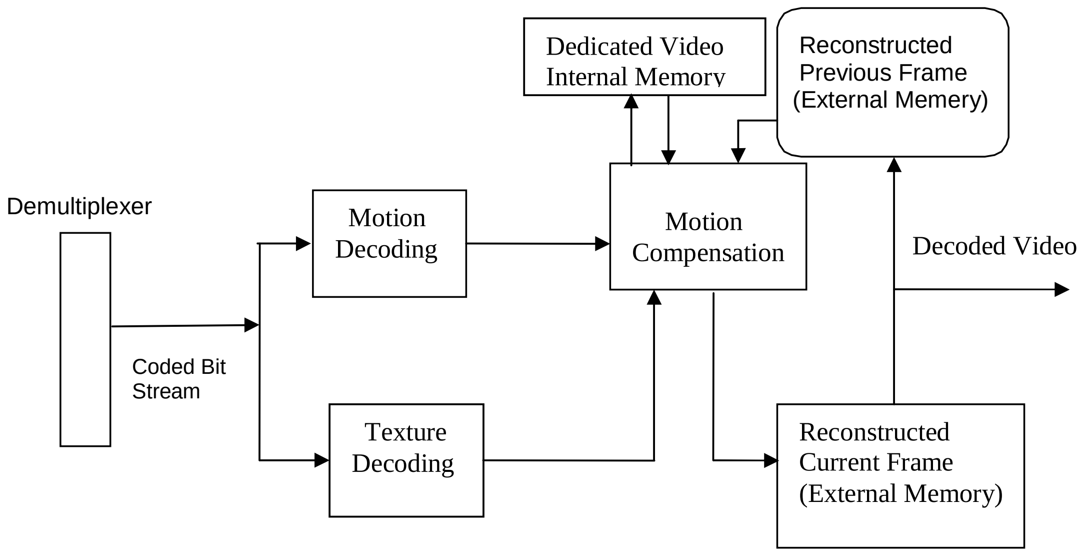
Dedicated Video Internal Memory
Reconstructed Previous Frame (External Memery)
Demultiplexer
Motion Decoding
Motion Compensation
Decoded Video
Coded Bit Stream
Texture Decoding
Reconstructed Current Frame (External Memory)

Fig. 2. A Video Decoder with Dedicated Internal Memory Catch for Storing Reference Blocks

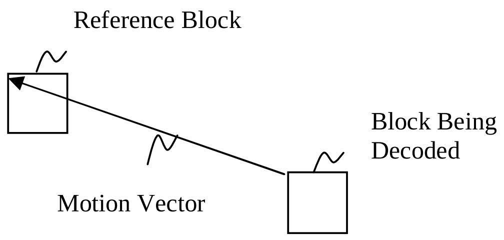

Fig.3a. The reference block specified by a motion vector.

Block Index	In /Out	Where in DVIM

Fig. 3b. The look-up table for DVIM..

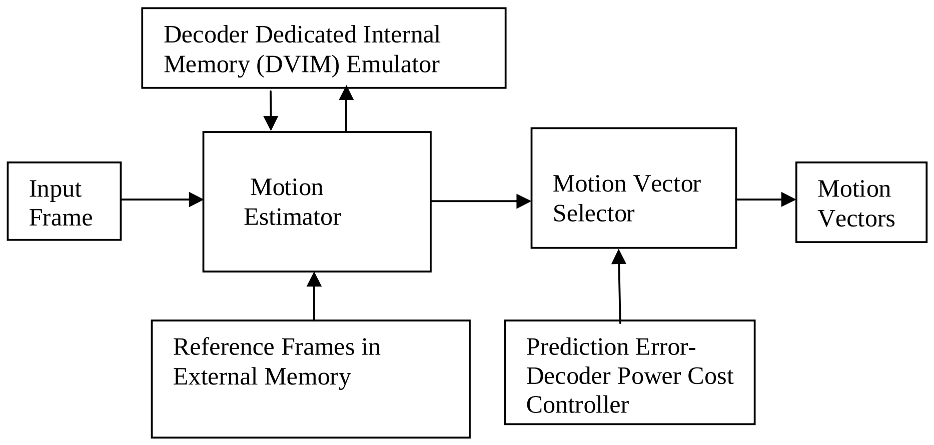
Decoder Dedicated Internal Memory (DVIM) Emulator
Input Frame
Motion Estimator
Motion Vector Selector
Motion Vectors
Reference Frames in External Memory
Prediction Error-Decoder Power Cost Controller

Fig. 4. Motion Estimation with Decoder Dedicated Internal Memory Emulator and Prediction Error-Decoder Power Cost Controller