LETTER

Extended MPEG Video Format for Efficient Dynamic Voltage Scaling

Kwanhu BANG[†], **Sung-Yong BANG**[†], ***Nonmembers***, ***and*** **Eui-Young CHUNG**[†a)], ***Member***

SUMMARY We present an extended MPEG video format for efficient Dynamic Voltage Scaling (DVS). DVS technique has been widely researched, but the execution time variation of a periodic task (i.e. MPEG decoding) is still a challenge to be tackled. Unlike previous works, we focus on the data (video stream) rather than the execution code to overcome such limitation. The proposed video format provides the decoding costs of frames to help the precise prediction of their execution times at client machines. The experimental results show that the extended format only increases the data size less than 1% by adding about 10 bits representing the decoding cost of each frame. Also, a DVS technique adjusted for the proposed format achieves 90% of efficiency compared to the oracle case, while keeping the run time overhead of the technique negligible.

key words: *low power, energy, DVS, video, decoding, MPEG*

1. Introduction

In ubiquitous era, the demand for portable computing and communication devices has been rapidly increasing. One of their most important metrics is power consumption. ITRS (International Technology Roadmap for Semiconductors) predicts that portable devices implemented with low operating power (LOP) process technology will consume power 8 times more than the required specification without any power management scheme in 2018 [1]. For this reason, the power management scheme is an essential feature of contemporary portable devices and many of them are tailored to the video application, since it is one of the most popular applications of portable devices. Dynamic Voltage Scaling (DVS) technique is one of the most effective power management schemes and has been extensively researched for video decoding. Simply speaking, DVS is a technique to schedule voltage/frequency pairs for a task (or tasks) to be executed without violating the given deadline. In case of video decoding, we can model it as a single task and its execution time varies depending on the frame characteristics. Also, there are several real time constraints such as frame decoding rate. The common challenge of DVS technique for video applications is how precisely estimate the decoding time of the given video stream, since the exact estimation of an idle period (i.e. the difference between the deadline and the execution time) can be utilized for an optimal voltage/frequency scheduling. In this letter, we only focus on the DVS techniques developed for video applications, especially for MPEG video decoding.

Manuscript received August 27, 2007.
Manuscript revised December 14, 2007.
[†]The authors are with Yonsei University, Seoul, Korea.
a) E-mail: eychung@yonei.ac.kr
DOI: 10.1093/ietfec/e91–a.5.1283

Several DVS techniques for MPEG video decoding were proposed. In [2] and [3], they predicted the decoding time of a frame based on the ratio of idle period and busy period experienced in the recent past. On the other hand, authors in [4] predicted the decoding time using the encoded data size. In [5], they predicted the decoding time in the unit of group of pictures (GOP) which consists of several frames using the frame types and sizes. Authors in [6] predicted the decoding time in further fine-grained level. They partitioned the execution code of a frame decoding into several steps and classified them into two categories — frame-dependent and frame-independent. They predicted the decoding time only for the frame-dependent steps, while considering a constant decoding time for each frame-independent step. They further improved their technique in [7] by decomposing the operations into memory bound operations and CPU bound operations using a performance monitoring unit (PMU). Such decomposition further exploited idleness for the memory bound operations. Many of them suffered from the prediction accuracy, since they predicted the decoding time based on the recent history or indirectly related parameters.

Unlike these works, authors in [8] focused on the data (video stream) itself rather than the execution code (i.e. video decoder). In other words, they required the contents providers to deliver the decoding time information with the corresponding video stream. The basic rationale of this method is to reduce the uncertainty of decoding time which is the major obstacle of other previous works. Even though the contents providers should perform more work to provide such information, many end users (from a few to millions depending on the popularity of the contents) can be benefited by saving the energy consumed by their portable devices. However, the technique requires the modification of the MPEG decoder to utilize such information for DVS and raises the compatibility issue. Also, they did not address any implementation details to combine the decoding time information with the video stream.

In this letter, we propose an extended MPEG format which complies with the technique proposed in [8]. Notice that the newly introduced attributes in the extended MPEG format are specified in the user-defined fields, thus the video stream with new attributes can be still decoded by the original MPEG decoder by discarding these attributes. The proposed format provides more information for efficient DVS, while keeping the backward compatibility with the original MPEG decoder.

The remainder of this letter is organized as follows. Target DVS technique is described in Sect. 2. In Sect. 3, an extended MPEG format for DVS is presented. In Sect. 4, an extended MPEG decoder for the target DVS technique is explained. Simulation environment and Experimental results are shown in Sect. 5. Finally, conclusions are given in Sect. 6.

2. Target DVS Technique

This section is devoted to the summary of the technique proposed in [8] which is our target DVS technique to show the effectiveness of video format extension. We simply call this technique the CP-DVS (Contents Provider-assisted DVS) in this letter. Even though the proposed format extension can be incorporated with any other DVS techniques, we focus on the technique, since authors in [8] clearly specified the necessary attributes for their techniques. Notice that our objective is not to propose a new DVS technique, but to propose an extended video format that carries more information necessary for the precise execution time prediction.

CP-DVS represents the decoding time of a single frame in a relative manner. More precisely, it normalizes the execution time (decoding time) of each frame to that of the first frame. This helps to decouple the decoding time from the architecture-dependent characteristics by assuming that the decoding time variation according to the architecture changes can be modeled as a simple piece-wise linear model. This is very important, since many client machines (end users) will have different architectures from the reference machine of the contents provider. In other words, the linear scaling of decoding time would be enough to overcome the various architecture types. For this purpose, CP-DVS defines "decoding cost" which is the decoding time of each frame normalized to that of the first frame. It also defines "best decoding cost" and "worst decoding cost" which are the smallest and the largest decoding cost among all the frames in the given video stream, respectively.

The decoding cost, best decoding cost and worst decoding cost are obtained during the characterization process by the contents provider at the reference machine. The contents provider should have a customized MPEG encoder which adds those parameters to the given video stream. The best decoding cost and worst decoding cost are selected from the entire video frames, while the decoding cost is given to each frame.

The client machine also needs an MPEG decoder customized for CP-DVS. It first builds two tables called scaling table and DVS table, respectively. The number of rows in each table is the difference of worst decoding cost and best decoding cost divided by the unit of resolution which trades off the size of table and the accuracy of execution time prediction. The scaling table translates the decoding cost at the reference machine into the decoding cost at the client machine (called actual decoding cost) by defining a parameter called scaling factor. It is the ratio of the decoding cost at the client machine over the decoding cost at the reference machine. The actual decoding cost is the product of the scaling factor and the decoding cost given by the frame to be decoded. The scaling factor of each row at the scaling table is initially unknown. Whenever each frame decoding is started, the row corresponding to its decoding cost is selected to check its scaling factor is unknown (the Check Phase). If it is already known, DVS table is used to select optimal voltage/frequency pair (the DVS Phase). Otherwise, CP-DVS MPEG decoder decodes the frame at the full speed and measures the decoding time, then it computes the scaling factor by the ratio of the measured value over the decoding time of the first frame at the client machine (the Learning Phase). Each row of the DVS table has three entries — voltage, frequency, and threshold. The threshold value means the corresponding voltage and frequency pair is optimal when the actual decoding cost is larger than the threshold value. Further details can be found at [8].

3. Extended MPEG Format

Figure 1 shows the hierarchy structure of MPEG-2 video format [9]. As shown in Fig. 1, MPEG-2 video format consists of four-level hierarchies. Each level has a header field which includes the necessary information for decoding. Also, each level has an extension field and/or a user data field. These fields can be utilized for storing the additional information such as decoding cost for DVS. Notice that we can choose the levels depending on the controlling granularity of DVS. In this work, we utilize the sequence level and picture level to control the voltage/frequency pairs on a per-frame basis using CP-DVS. However, other fields can be also utilized with other DVS techniques if necessary. For instance, GOP-level header or user data field can be utilized in corporation with the technique in [5]. It is also worth to mention that the similar extension is possible with video formats such as H.264 which provides a higher compression rate than MPEG-2, since most of recent video formats are similar to the structure shown in Fig. 1. Note that the sequence header (shaded by the gray color in Fig. 1) should be extended to keep the global information without any option, while the GOP-level header, picture-level header or slice-level header (dotted by the gray color in Fig. 1) can be selectively extended depending on the granularity of the target

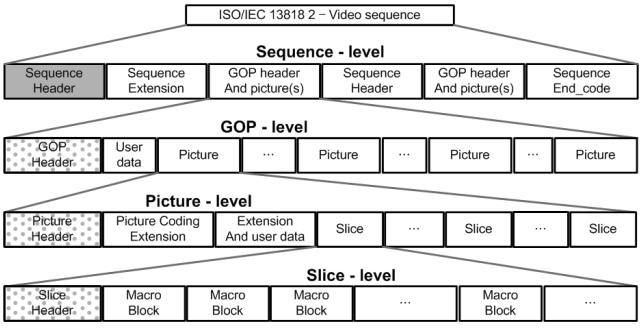

Fig. 1 The hierarchy structure of MPEG-2 video format.

Table 1 The extended attributes of sequence header.

Sequence header field	Number of bits
sequence_header_code	32
horizontal_size_value	12
vertical_size_value	12
aspect_rate_code	4
.....	...
decoding_time_first	9 and above
decoding_cost_best	9 and above
decoding_cost_worst	9 and above

Table 2 The extended attribute of picture header.

Picture header field	Number of bits
picture_start_code	32
temporal_reference	10
picture_coding_type	3
vbv_delay	16
.....	...
current_decoding_cost	9 and above

DVS policy. CP-DVS chooses the picture-level to schedule optimal voltage and frequency pair per-frame basis.

The extended fields and user data are declared as pointer variables in MPEG-2 data structure. These pointer variables are initially assigned to NULL, thus original MPEG-2 decoder ignores these fields when they are decoding a given video stream. In our case, we allocate several bits to these pointer variables to capture the information necessary for CP-DVS. The number of bits assigned to each field will trade off the increase of video data size and accuracy. We will show the trade-off analysis in Sect. 5. Table 1 and Table 2 show the sequence header and picture header including the extended attributes, respectively. For sequence header, we allocate three attributes — the first frame decoding time (decoding_time_first in Table 1), the best decoding cost (decoding_cost_best in Table 1), and the worst decoding cost (decoding_cost_worst in Table 1). Also, we allocate a single attribute called current_decoding_cost to store the decoding cost of the corresponding frame. The reason that the number of bits for new attributes are determined as "9 and above" will be explained in Sect. 5. The MPEG decoder should be appropriately modified to understand these attributes and include the target DVS policy.

Note that the header extension does not cause any compatibility issue with the MPEG decoders which support the original MPEG video format, since the extended fields are just ignored by these decoders.

4. Extended MPEG Decoder

We extended the open source MPEG codec from [10] and [11] to implement the CP-DVS with the proposed format. Figure 2 shows the pseudo-code of the MPEG decoder customized for CP-DVS. The underlined codes are inserted to the original decoder for this purpose. Initialization is performed by function mpeg2Init() on line 3. In addition to the initializations done by original MPEG decoder, out implementation sets the processor with its maximum voltage and frequency. Then, the extended code performs the decoding iteratively from line 6 to line 58. We will describe each code section inside the while loop more in detail.

```
1 CPDVS (SCALINGTABLE *scalingTable, DVSTABLE *dvsTable) {
2   /* initialization of decoder */
3   mpeg2Init();
4
5   /* MPEG-2 decoding loop */
6   while(!VIDEO_END) {
7     /* read video data from video clip */
8     readVideoData(buffer, fileVideoClip);
9
10    /* MPEG-2 header decoding */
11    if(SEQUENCE_HEADER_CODE) {    // for sequence header
12      getSequenceHeaderInfo();
13      getSequenceCostInfo();
14      /* Initialization Phase */
15      constructScalingTable(scalingTable, worstCost, bestCost);
16    }
17    if(GOP_HEADER_CODE) {         // for GOP header
18      getGOPHeaderInfo();
19    }
20    if(PICTURE_HEADER_CODE) {     // for picture header
21      getPictureHeaderInfo();
22      getPictureCostInfo();
23
24      if(!FIRST_FRAME) {
25        if((scalingFactor = checkPhase(pictureCost))>0) {
26          /* DVS phase */
27          NEED_LEARNING = FALSE;
28          DVSPhase(scalingFactor, pictureCost);
29        } else {
30          NEED_LEARNING = TRUE;
31          DVS(MAX_SPEED);
32        }
33      }
34    }
35
36    /* measure decoding time */
37    startTime = getTime();
38    while (!PCITURE_END){
39      if(SLICE_HEADER_CODE) {    // for slice header
40        getSliceHeaderInfo();
41      }
42      mpeg2BlockDecoding(buffer);
43    }
44    endTime = getTime();
45    frameDecodingTime = endTime - startTime ;
46
47    if(FIRST_FRAME) {
48      /* Compulsory learning phase */
49      compulsoryLearningPhase(scalingTable, frameDecodingTime);
50      constructDVSTable(dvsTable, frameDecodingTime);
51      firstFrameDecodingTime = frameDecodingTime;
52    } else {
53      if(NEED_LEARNING == TRUE)
54        /* Learning phase */
55        learningPhase(frameDecodingTime);
56    }
57    frameDecodingTime = 0;
58  }
59 }
```

Fig. 2 The pseudo-code of the extended MPEG decoder.

4.1 Header Parsing

This part is from line 7 to line 22 in Fig. 2. First, CP-DVS reads video data (from line 7 to line 8). Then, CP-DVS reads the sequence header including the extended attributes described in Sect. 3 (from line 11 to line 13). After that, the scaling table mentioned in Sect. 2 is constructed using these attributes (at line 15). Then, CP-DVS iteratively reads

each GOP header and each picture header nested in a GOP (from line 17 to line 22). The GOP header has nothing for CP-DVS because we extended the MPEG format per-frame basis. The attribute "current_decoding_cost" mentioned in Sect. 3 is obtained from each picture header (at line 22).

4.2 Voltage and Frequency Setting

This part is from line 24 to line 34 in Fig. 2. For the first picture header, CP-DVS skips this part and and starts frame decoding. Note that the processor is already set to its maximum voltage and frequency at the initialization phase. In the other cases, CP-DVS first checks there is an applicable scaling factor for the current decoding cost (at line 25). If the scaling factor exists, CP-DVS unsets a flag called "NEED_LAERNING" for the learning phase and adjusts the voltage and the frequency with the corresponding DVS table information (from line 26 to line 28). The function "DVSPhase()" on line 28 performs the voltage and frequency setting in our implementation. If the scaling factor does not exist, the flag is set and CP-DVS sets the processor with the maximum speed (from line 30 to line 31).

4.3 Actual Frame Decoding Time Measurement

This part is from line 36 to line 45 in Fig. 2. After CP-DVS reads a picture header, it decodes the corresponding frame while measuring the decoding time of the frame (from line 36 to line 45). The measured time is used to update the scaling table and the DVS table for the compulsory learning phase and learning phase at each end of frame decoding.

4.4 Learning Phase

This part is from line 47 to line 57 in Fig. 2. When the first frame is decoded, the compulsory learning phase is performed (at line 49), and the DVS table is constructed using the actual decoding time of the first frame (at line 50). Next, the time is saved to calculate the scaling factor (at line 51). For the other frames, CP-DVS checks the flag for the learning phase and performs the learning phase if "NEED_LEARNING" has been set (from line 53 to line 55). At the end, the variable for the actual decoding time is reset to zero to be used at next frame (at line 57).

5. Experimental Results

We conducted several sets of experiments using a cycle-accurate simulator called MaxSim from ARM [12]. MaxSim provides not only the simulation kernel, but also simulation libraries including ARM processors. For each set of the simulation, we used the video clips listed in Table 3. Each video clip consists of 60 frames.

We first focus on the bit-width optimization of the new attributes introduced by the extended video format. To avoid the heavy computations of floating-point numbers, we implemented the CP-DVS algorithm in the extended MPEG decoder by using the fixed-point operations. Figure 3 shows the computation accuracy for selecting optimal voltage and frequency pair as a function of bit-width when we use the fixed-point operations against the floating-point operations. As shown in Fig. 3, 9-bit is enough for each attribute to attain the reasonable accuracy (higher than 90%).

Next, we evaluated the computational overhead of CP-DVS technique with the extended format when the bit-width of each new attribute is 13. The results are shown in Table 4 with the size increase of each video clip due to the new attributes. From Table 4, it is obvious that the proposed video format slightly incurs both the computational overhead and the size overhead, meaning that it satisfies the most essential properties required to the DVS techniques. Also, both computational and size overheads decrease as the picture resolution increases, because the number of attributes is unchanged while the data size and its decoding time increase.

Finally, we measured the energy saving achieved by the proposed video scheme with CP-DVS and the results are shown in Fig. 4. We used the power numbers, voltage/frequency pairs, and delays provided by [13]. As shown in Fig. 4, CP-DVS with the extended MPEG video format

Table 3 Video clips used for simulations.

Video clip	Resolution 1	Resolution 2
bigbend	QSIF (160 × 120)	SIF (320 × 240)
linda	QSIF (160 × 120)	SIF (320 × 240)
game_pov	QSIF (160 × 120)	SIF (320 × 240)
museum	QSIF (160 × 120)	SIF (320 × 240)
elevator	QSIF (160 × 120)	SIF (320 × 240)
RedsNightmare	QSIF (160 × 120)	SIF (320 × 240)

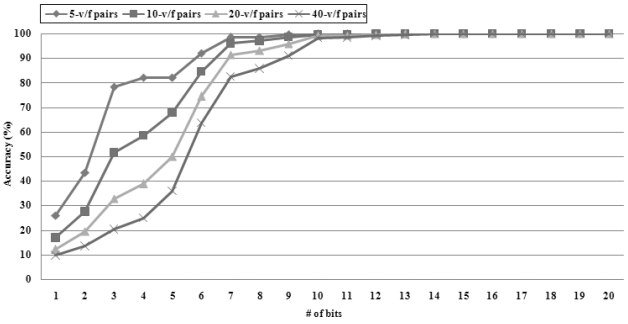

Fig. 3 The computation accuracy for selecting optimal voltage and frequency pair as a function of bit-width.

Table 4 Computational overhead and size increase by the extended MPEG format.

Overhead	Computational overhead		Size increase	
Video clip	Resolution		Resolution	
	QSIF	SIF	QSIF	SIF
bigbend	1.4069%	0.5875%	0.0017%	0.0003%
linda	1.1515%	0.1818%	0.0208%	0.0068%
game_pov	1.1042%	0.4336%	0.0058%	0.0005%
museum	0.9186%	0.3447%	0.0038%	0.0013%
elevator	1.0120%	0.3699%	0.0036%	0.0010%
RedsNightmare	1.0440%	0.2986%	0.0219%	0.0075%

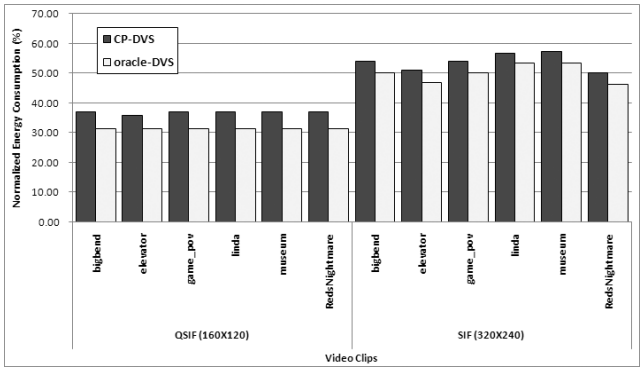

Fig. 4 Comparison of normalized energy consumption between CP-DVS and oracle-DVS.

consumed about 60%(50%) less energy than the original MPEG decoder for QSIF (SIF). Also, it is about 90% of efficiency compared to the oracle DVS. We achieved the energy saving similar to the results reported from [8], meaning that the proposed video format is well implemented without any major overhead and enables CP-DVS to behave like the oracle DVS.

6. Conclusions

In this letter, we proposed an extended MPEG video format which can be incorporated with CP-DVS or any other DVS techniques. We define the new attributes in the extended format and their optimal bit-width is proposed to minimize the computational overhead by using the fixed-point operations. The experimental results show that the proposed method achieves the energy saving similar to the results reported from CP-DVS, while the overheads caused by the proposed method are negligible. Finally, the proposed technique can be applicable to other video formats for DVS.

Acknowledgments

This work was partially supported by the Korea Science and Engineering Foundation (KOSEF) grant funded by the Korea government (MOST) (No. R01-2006-000-10156-0) and by IDEC (IC Design Education Center).

References

[1] http://public.itrs.net
[2] T. Pering, T. Burd, and R. Broderson, "The simulation and evaluation of dynamic voltage scaling algorithms," Proc. Int'l Symp. on Low Power Electronics and Design, pp.76–81, 1998.
[3] D. Grunwald, P. Levis, K. Farkas, C. Morrey, III, and M. Neufeld, "Policies for dynamic clock scheduling," Symp. on Operating Systems Design & Implementation, pp.73–86, Oct. 2000.
[4] J. Pouwelse, K. Langendoen, R. Lagendijk, and H. Sips, "Power-aware video decoding," 22nd Picture Coding Symp., Seoul, Korea, 2001.
[5] D. Son, C. Yu, and H. Kim, "Dynamic voltage scaling on MPEG decoding," Int'l Conf. of Parallel and Distributed System, pp.633–640, June 2001.
[6] K. Choi, K. Dantu, W. Cheng, and M. Pedram, "Frame-based dynamic voltage and frequency scaling for a MPEG decoder," Proc. Int'l Conf. on Computer Aided Design, pp.732–737, Nov. 2002.
[7] K. Choi, R. Soma, and M. Pedram, "Off-chip latency-driven dynamic voltage and frequency scaling for an MPEG decoding," Design Automation Conference, pp.544–549, June 2004.
[8] E.-Y. Chung, L. Benini, and G. De Micheli, "Contents provider-assisted dynamic voltage scaling for low energy multimedia applications," Proc. 2002 International Symposium on Low Power Electronics and Design, pp.42–47, 2002.
[9] ISO/IEC 13818-2, "Generic coding of moving pictures and associated audio," 1993.
[10] http://www.mpeg.org/MPEG/MSSG
[11] http://libmpeg2.sourceforge.net
[12] http://www.arm.com/products/DevTools/MaxSim.html
[13] Intel Corporation, "Intel® PXA27x Processor Family Power Requirements," 2004.