Chie Dou, Yong-Lin Lee/ International Journal of Engineering Research and Applications (IJERA)
ISSN: 2248-9622 www.ijera.com
Vol. 2, Issue 3, May-Jun 2012, pp.1643-1653

Frame Dropping Algorithms for Dynamic Voltage Scaling in Battery-Aware Low-Power Video Decoding

Chie Dou, Yong-Lin Lee

National Yunlin University of Science and Technology
No. 123, Section 3, University Road
64002 Touliu, Yunlin, Taiwan

Abstract. Dynamic voltage scaling (DVS) is an effective technique for low-power video decoding in battery-aware mobile devices. Once a frame is decoded, the processor waits until its deadline, when the frame must be played out. This waiting time is used to exploit the advantage of DVS. However, the predicted decoding time of a frame may exceed its deadline (or called expected decoding time, *EDT*). The decoding of a frame may be deferred or even the frame may be dropped when frames with large predicted decoding times occurred in a burst. This paper proposes two types of frame dropping algorithms to maintain the energy efficiency of DVS in MPEG decoding; meanwhile, to avoid consecutive frame dropping to preserve the quality of video playing. The difference between these two algorithms is determined by when and how to drop a frame. The first algorithm, called *Late_dropping* algorithm, selects a frame to drop when the deferred decoding time of a frame is larger than one *EDT*. And the second algorithm, called *Early_dropping* algorithm, selects a frame to drop when the deferred decoding time of a frame is larger than one half of an *EDT*. Performance comparisons between these two algorithms have been investigated, using a heavy burst interval in a sample video clip as an evaluation basis. Although *Early_dropping* algorithm produces almost twice more dropping situations than that of *Late_dropping* algorithm, it acquires better DVS application and thus saves more battery energy to about 20% than that of *Late_dropping* algorithm. Evaluation results also show that for both algorithms frame dropping will never be occurred in consecutive.

Keywords: Dynamic Voltage Scaling, battery energy, mobile devices, MPEG

1 Introduction

As computation requirement of modern multimedia applications increases, efficient energy-saving techniques become more important for the battery-aware mobile devices. Dynamic Voltage Scaling (DVS) allows a processor to dynamically change its voltage and speed at run time, thus increasing energy efficiency. MPEG video playing is one of the most popular killer applications in mobile environment. It requires considerable computing power (battery energy); hence it is worthwhile to adapt DVS on it. The workload of MPEG decoding varies widely due to that MPEG video stream contains different frame types (*I*, *P*, *B*), and the wide variation between scenes, and it should confirm to the real-time constraints.

The system configuration for the battery-operated processor under consideration was illustrated in [1]. The system block diagram is depicted in Fig. 1. The system consists of one DVS processor driven by a single battery. The processor is powered by the battery through a DC-DC converter. In Fig. 1, I_k denotes the current drawn from the battery during execution of task *k*, and P_k denote the power demanded by task *k* from the output of the DC-DC converter. Assume that the battery voltage is some averaged constant, and then we have $I_k \propto P_k$. Also, let V_k and $\emptyset_k$ denote the operating voltage and the maximum clock frequency for task *k*, respectively. Since $P_k \propto V_k^2 \emptyset_k$ and $\emptyset_k$ is approximately proportional to V_k, we may have the following approximation: $I_k \propto V_k^3$. In addition, the task delay Δ_k is obtained as $\Delta_k = N_k / \emptyset_k$, where N_k is the number of clock cycles necessary to complete task *k*, and so that $\Delta_k \propto V_k^{-1}$.

Fig. 1 System block diagram.

MPEG Encoding/Decoding

A standard MPEG stream is composed of three types of compressed frames: *Intra-coded (I)*, *Predictive-coded (P)*, and *Bidirectional-coded (B)*. On the average, *I* frames are the largest in size, followed by *P* frames, and finally *B* frames. Since MPEG video has a regular pattern on the sequence of frame types, called GOP (group of pictures), such as "*IBBPBBPBB*", the decoding time of the video sequence conforms to a regular pattern. We adopt a video clip from a high motion movie

"*UnderSiege"* used in [2] for evaluating our proposed DVS mechanism and frame dropping algorithm. Main characteristics of MPEG stream in this *UnderSiege* clip are shown in Table 1 below.

Table 1. *UnderSiege* movie clip characteristics.

Type	Movie
Frame Rate	30 fps
I frames	122
P frames	122
B frames	486

We also adopt the linear regression model used in [2] to describe the linear relationship between the two metrics, the frame size and the number of clock cycles required to decode the frame. Because the decoding clock cycles per byte are different for different frame types. On the average, *I* frame requires less decoding clock cycles per byte while *B* frame requires more decoding clock cycles per byte. Table 2 shows the regression model for the expected decoding clock cycles with an R^2 coefficient of the linear regression through the sample points.

Table 2. Regression model for the expected decoding clock cycles.

Frame type	Regression model	R^2
I frame	$39 \times \text{frame size} + 4.7 \times 10^6$	0.96
P frame	$64 \times \text{frame size} + 1.9 \times 10^6$	0.92
B frame	$115 \times \text{frame size} + 1.1 \times 10^6$	0.95

Dynamic Voltage Scaling (DVS)

Although it is generally desirable to have higher frequency for faster instruction execution, for some tasks maximum execution speed is not required and so that the supply voltage and operating frequency can be reduced. DVS allows a processor to dynamically change its frequency and voltage at run time, increasing energy efficiency. For real-time embedded systems, the execution of tasks can be slowed down to save processor energy as long as the deadline constraints are not violated. For MPEG decoding, the processor runs at a maximum speed when decoding, i.e., the highest voltage/frequency setting, to avoid deadline violation. Once a frame is decoded, the processor waits until its deadline (e.g., every 33.3 ms for the frame rate of 30 fps), when the frame must be played out. It may be regarded that during this waiting time the processor is idle. These idle periods are used to exploit the advantage of DVS. In this paper, we assume ideal DVS is adapted on MPEG decoding, where the processor scales down exactly to the voltage/frequency setting required to decoding the corresponding frame. Such that no idle time exists and power saving is maximized. In Fig. 1, we have shown that $I_k \propto V_k^3$, and $\Delta_k \propto V_k^{-1}$ as well. That is to say, I_k will be decreased inversely proportional to the third power of Δ_k [1],[3],[4].

The rest of the paper is structured as follows. Section 2 gives a brief introduction to different battery models and a short overview of diffusion model that is used in our study. In section 3, a probabilistic approach for selecting suitable maximum decoding frequency (MDF) for different frame types in MPEG decoding is presented. Section 4 proposes two types of frame dropping algorithms to maintain the energy efficiency of DVS in MPEG decoding when frames with large predicted decoding times occurred in a burst; meanwhile, to avoid consecutive frame dropping to preserve the quality of video playing. In section 5, performance comparisons both on frame dropping and energy saving between these two algorithms were investigated, using a heavy burst interval in a sample video clip as an evaluation basis. Section 6 concludes the paper with an outlook to future works.

2 Battery Model

Many different types of battery models have been developed for different application areas [5],[6],[7]. For example, the electrochemical models are based on the chemical processes that take place in the battery. In electrical-circuit models the electrical properties of the battery are modeled using PSpice circuits. The stochastic models describe the battery based on discrete-time Markov chains at a higher level of abstraction than the electrochemical and electrical circuit models. The Kinetic battery model (KiBaM) and the diffusion model are two well-known analytical models. Both models are well

suited for performance evaluation purposes and for doing battery lifetime predictions. In this paper, we use the diffusion model [1],[5],[8] which is a parametrically simple analytical form that relates the battery lifetime to the time-varying load profile.

Let D denote the diffusion constant of the diffusion model and w be the width of the linear diffusion region. An expression for the apparent charge lost from the battery, represented by $\alpha(t)$, can be obtained as follows [5].

$$\alpha(t) = \int_0^t i(\tau)d\tau + \int_0^t i(\tau)(2\sum_{m=1}^{\infty} e^{-\beta^2 m^2 (t-\tau)})d\tau \quad (1)$$

where $\beta = \pi\sqrt{D}/w$. The charge lost can be divided into two parts: the first is the charge consumed by the task, and the second is the charge which remains unused in the battery. The battery is vacant when the charge lost is equal to the battery capacity. For a constant current ρ, the function that describes the restoration of the unavailable charge during an idle period after a load ρ that lasted for a period of length t_l is [5]

$$u(t_i) = 2\rho \sum_{m=1}^{\infty} \frac{e^{-\beta^2 m^2 t_i}(1 - e^{-\beta^2 m^2 t_l})}{\beta^2 m^2}, \quad (2)$$

where t_i is the idle time.

Battery Load Profile

Fig. 2 shows a sequence of tasks each of which charges a constant load on the battery. Let I_k, Δ_k, and t_k denote the current, duration, and the start time of task k, respectively. Assume that the battery has a total discharge time T, and the resulting load profile is an n-step staircase function. The load profile is specified by the three set [5]: the current set $S_I = \{I_k | k = 0, 1, \ldots, n-1\}$; the duration set $S_\Delta = \{\Delta_k | k = 0, 1, \ldots, n-1\}$; and the start time set $S_t = \{t_k | k = 0, 1, \ldots, n-1\}$. Then, the battery charge consumed by task k, represented by α_k, is obtained as follows.

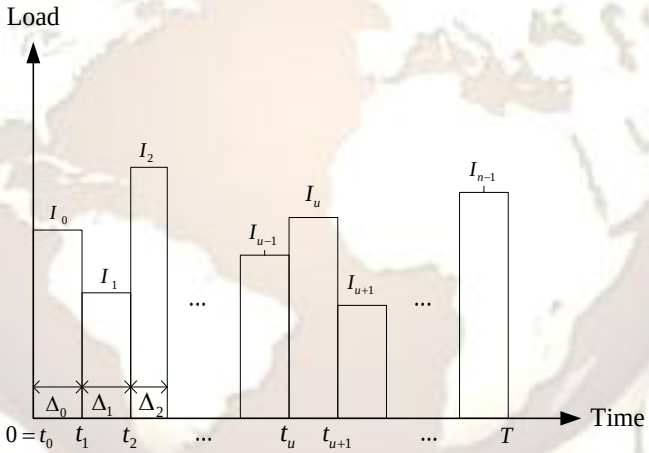


Fig. 2 Battery load profile.

$$\alpha_k = I_k \times F(T, t_k, t_k + \Delta_k, \beta), \quad (3)$$

where

$$F(T, t_k, t_k + \Delta_k, \beta) =$$

$$\Delta_k + 2\sum_{m=1}^{\infty} \left[\left(e^{-\beta^2 m^2 (T - t_k - \Delta_k)} - e^{-\beta^2 m^2 (T - t_k)} \right) \Big/ (\beta^2 m^2) \right]. \quad (4)$$

3 Maximum Decoding Frequency Setting

We apply off-line speed determination by treating the execution of each MPEG frame as its best-case estimation and adopt slack reclamation to reduce the energy consumption in an on-line (run-time) manner. Off-line speed determination can be applied to many popular applications, such as video-on-demand, movie-on-demand, and multimedia-on-demand. So that, the predicted decoding time for each frame can be estimated in advance in an off-line manner. To determine the off-line "suitable maximum" decoding frequency for different frame types individually, a probabilistic approach is proposed as follows. Fig. 3(a) shows the probability mass function (pmf) of the frame size for all *I* frames in the *UnderSiege* movie clip. Shown in Fig. 3(b) is the corresponding predicted decoding time versus frame size under different decoding frequencies.

We assume frame rate is 30 fps, that is, the deadline (or expected decoding time) of each frame is 33.3 ms. Fig. 3(b) also uses rectangles to indicate the maximum frame size and the probability that the predicted decoding time exceeds 33.3 ms for each decoding frequency, respectively. From Fig. 3(b) we observed the predicted decoding times of all *I* frames are almost within the deadline when the decoding frequency is 260 MHz, and when the decoding frequency is reduced to 200 MHz the predicted decoding time of almost half the *I* frames will exceed their deadline. For MPEG decoding, the processor runs at a "suitable maximum" speed when decoding, i.e., the "highest" frequency setting, to avoid deadline violation. The selected "suitable maximum" speed should expolit the advantages of DVS for low-power video decoding; meanwhile, sustain the stability of system performance. Fig. 4 and Fig. 5 show the similar results for *P* and *B* frames, respectively. If we assume the probability of deadline violation cannot exceed 5%, we may determine the maximum decoding frequencies for *I*, *P* and *B* frames from Figs. 3, 4, and 5, respectively. Let $f_{I,max}$ denote the maximum decoding frequency for *I* frames, thus we have $f_{I,max}$= 240 MHz from Fig. 3(b). Also, we may have $f_{P,max}$= 200 MHz and $f_{B,max}$= 120 MHz from Fig. 4(b) and Fig. 5(b), respectively. If we increase the probability of deadline violation to about 20%, the maximum decoding frequency for each frame type is lowered to: $f_{I,max}$= 220 MHz, $f_{P,max}$= 160 MHz, and $f_{B,max}$= 90 MHz. Of course, these maximum decoding frequencies are generally much lower than the highest speed of the processor. The predicted decoding time for the k^{th} frame of type *I* is obtained as follows.

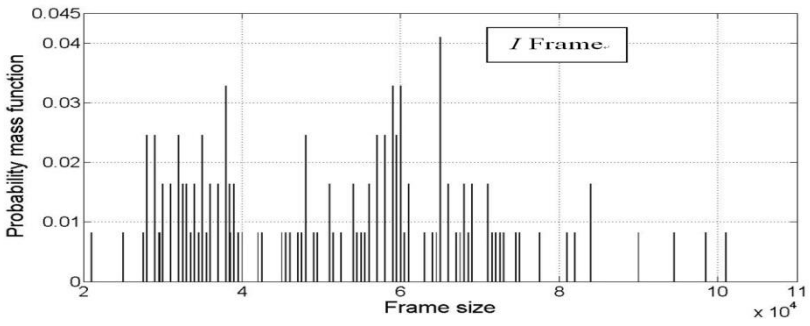
$$PDT_k^I = \frac{\text{The expected decoding clock cycles for the frame}}{f_{I,max}}. \quad (5)$$

Similar to (5), we may calculate the PDT_k^P and PDT_k^B for the k^{th} frame of type *P* and *B*, respectively. The slack time is reclaimed if the predicted decoding time is smaller than the deadline (expected decoding time, *EDT*) to slow down the execution speeds at run time to reduce the energy consumption. On the contrary, if the predicted decoding time is larger than the expected decoding time then a deadline violation is occurred. In the following section we propose a simple frame dropping algorithm cooperate with a buffer mechanism to minimize the effects of deadline violations on DVS in MPEG decoding.

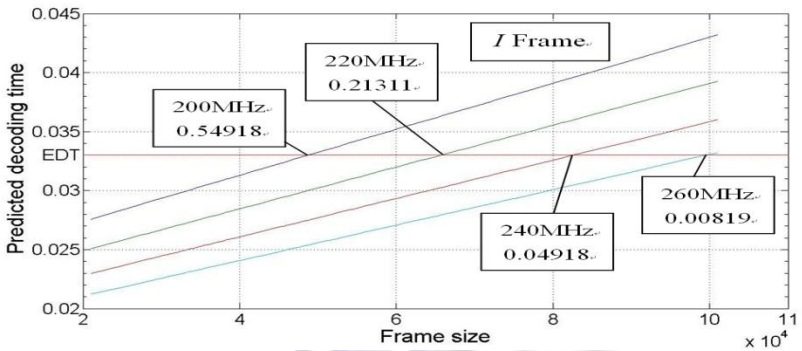
In this paper, we assume the on-line (run-time) initial discharge current I_{ini} for all types of frames is the same, 250 mA. Let ADT_k^I and I_k^I be the actual decoding time and discharge current, respectively, for the k^{th} frame of type *I*.
If $PDT_k^I < EDT$, we have $ADT_k^I = EDT$ (the slack time is reclaimed), and

$$I_k^I = I_{ini}\left(\frac{EDT}{PDT_k^I}\right)^{-3}. \quad (6)$$

Otherwise, if $PDT_k^I \geq EDT$ (deadline violation), we have $ADT_k^I = PDT_k^I$ and $I_k^I = I_{ini}$. Similar results can be applied to type *P* and *B* frames. Here we note that the above analysis is per frame basis, that is, we assume the decoding time of each frame starts from the beginning of the associated frame interval. Actually, when deadline violation occurs the decoding time of the next frame couldn't be started on schedule. Such delays of frame decoding will be accumulated if deadline violations occurred in a burst. We modify the above per frame basis analysis in the following section by considering the delay of frame decoding.

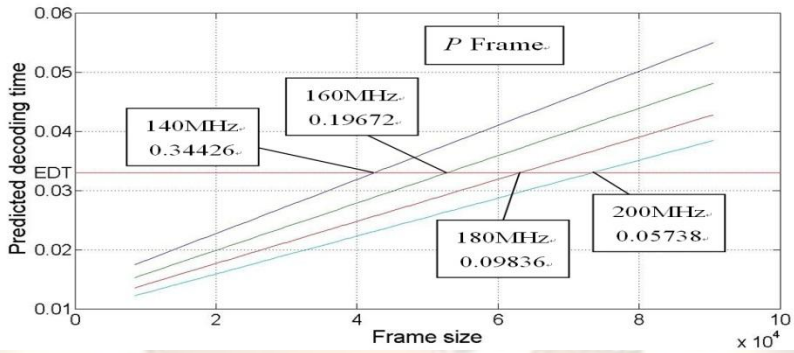


(a)

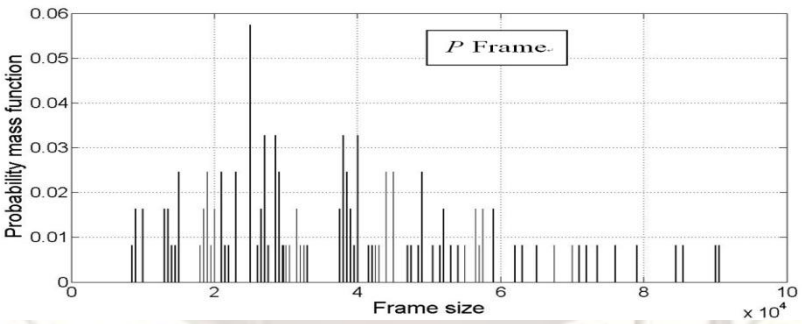


(b)

Fig. 3 Maximum decoding frequency setting for type-*I* frames in the *UnderSiege* movie clip.

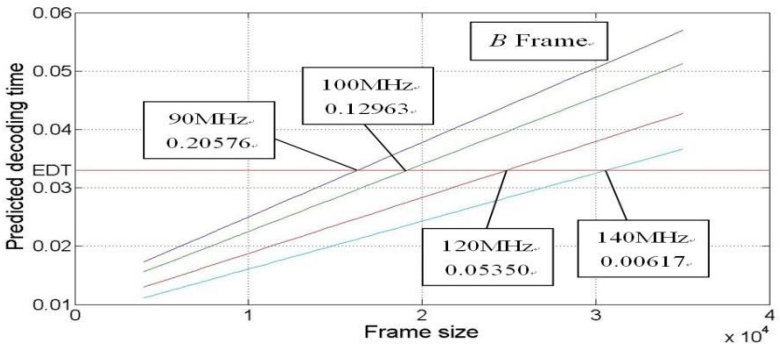


(a)

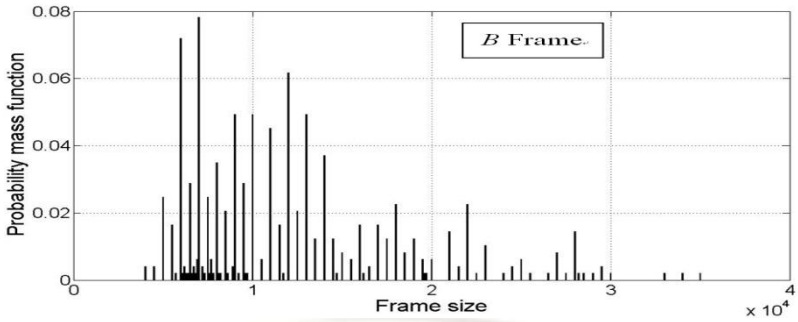


(b)

Fig. 4 Maximum decoding frequency setting for type-*P* frames in the *UnderSiege* movie clip.



(a)



(b)

Fig. 5 Maximum decoding frequency setting for type-*B* frames in the *UnderSiege* movie clip.

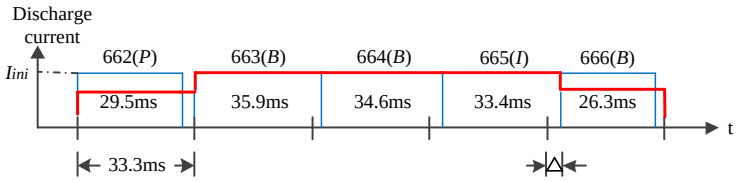
4 Frame Dropping Algorithms

Since we allow a frame to be decoded when deadline violation occurs, the decoding time of the next frame couldn't be started on schedule. Such delays of frame decoding will be accumulated if deadline violations occurred in a burst. In this paper, we use dual-buffer mechanism to solve the delay problem of frame decoding. One of the buffers is used to store the current decoding frame, and the other is used to store the next decoding frame. When the decoding time of a frame is deferred, the coded data of the frame can be stored in the buffer waiting for its decoding or to be dropped. Fig. 6 shows the battery discharge current of decoding frames versus time axis in the case of deadline violation probability is limited to 20%. The time axis is divided into equally spaced time interval which is equal to the expected decoding time (33.3 ms). In Fig. 6(a) and 6(b) we also show the frame type and the predicted decoding time for each frame. As shown in Fig. 6(a), frame number 662 is decoded on schedule and has no deadline violation, hence DVS is applied to this frame decoding and the discharge current follows equation (6). Frame number 663 is decoded on schedule but has deadline violation, hence DVS can't be applied to this frame and the decoding time of frame number 664 has to be deferred. Note that the decoding of frames 663, 664 and 665 all can't end before their respective deadlines, thus DVS can't apply to these frames and their discharge current are all equal to I_{ini}. Although the decoding of frame 666 has been delayed, the decoding of the frame ended before its expected deadline. Thus DVS can be applied to frame 666 and the discharge current needs to be modified as follows. Let $I_{deferred}$ and $PDT_{deferred}$ be the discharge current and the predicted decoding time of the frame such as frame 666, and let Δ be the deferred time of the frame decoding. We have

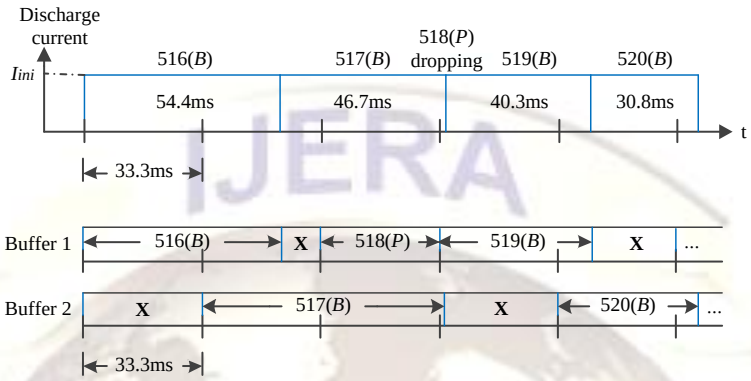
$$I_{deferred} = I_{ini}\left(\frac{EDT-\Delta}{PDT_{deferred}}\right)^{-3}. \quad (7)$$

Fig. 6(b) shows an example when frame dropping event occurred and the associated buffer content variations in both buffer 1 and buffer 2. In Fig. 6 (b), both frame 516 and 517 have large predicted decoding times and when the decoding of frame 517 is completed the decoding of frame 518 has just pass by. Thus the coded data of frame 518 is withdrawn from buffer 1 without being decoded. In Fig. 6 (b), we use crosses to indicate that during some time intervals the buffer contents are "don't care".

If we want to keep type *I* frame from being dropped to preserve video quality. In Fig. 6(b) it is necessary to make a decision what type does frame 518 belong to? If frame 518 belongs to type *I*, it shouldn't be dropped and instead of frame 519 it should be decoded immediately after the decoding of frame 517, so frame 519 is dropped without being decoded. Of course, the data content stored in buffer 1 has to be changed accordingly. Data frame 518 has to be stored in buffer 1 until the end of its decoding, and data frame 519 should not be appeared in buffer1. Actually, type *I* frames are most important frames in MPEG decoding and type *B* frames are least significant frames. Hence, instead of keeping type *I* frame from being dropped, the proposed frame dropping algorithms select only type *B* frames to drop in order to preserve video quality. The proposed algorithms also make a decision for each decoding frame whether DVS can be applied to it or not. For example, as shown in Fig. 6(b) the predicted decoding time for frame 520 is small than the *EDT*. If we consider the decoding of frame 520 individually, of course DVS should be applied on it. But, it didn't because of the delay of frame decoding. This will cause performance degradation due to deadline violations occurred in a burst.



(a) Decoding frame sequence without frame dropping.



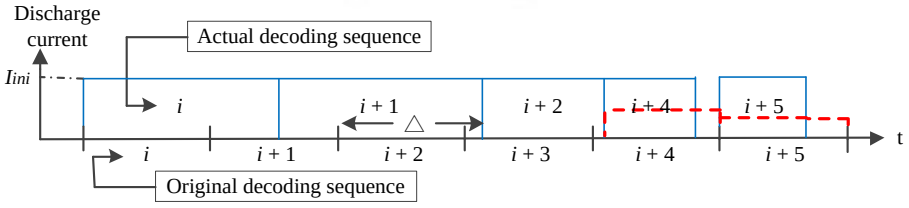
(b) Decoding frame sequence with frame dropping.

Fig. 6 Examples of decoding frame sequence with deadline violations occurred in a burst.

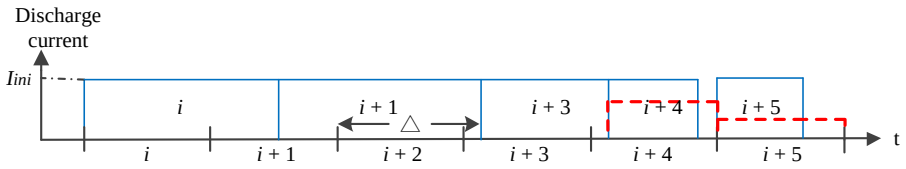
In this paper, we propose two types of frame dropping algorithms. The difference between these two algorithms is determined by when and how to drop a frame. The first algorithm, called *Late_dropping* algorithm, selects a frame to drop when the deferred decoding time of a frame, denoted by Δ, is larger than one *EDT*. The second algorithm, called *Early_dropping* algorithm, selects a frame to drop when the deferred decoding time of a frame is larger than one half of an *EDT*. In the following we use schematic diagrams to illustrate how to drop a frame for different algorithms. Schematic illustrations are helpful in understanding and differentiating when and how the proposed algorithms are going to drop a frame.

4.1 *Late_dropping* Algorithm

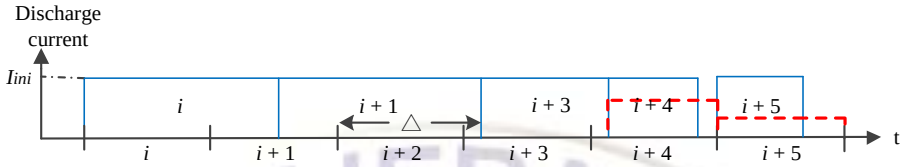
Fig. 7 shows the schematic illustrations for two kinds of *Late_dropping* algorithms by adopting different frame dropping considerations. Diagrams shown in Figs. 7(a) and 7(b) use drop_next mechanism, that is, if the deferred decoding time of $(i+2)^{th}$ frame is larger than one *EDT*, the algorithm allows the decoding of the $(i+1)^{th}$ frame to continue and selects next frame to drop. If the next frame, i.e., the $(i+2)^{th}$ frame, belongs to *I* or *P* type, then the algorithm selects the $(i+3)^{th}$ frame to drop as shown in Fig. 7(a). But if the $(i+2)^{th}$ frame belongs to *B* type, then the algorithm selects the $(i+2)^{th}$ frame to drop as shown in Fig. 7(b). Diagrams shown in Figs. 7(c) and 7(d) use immediately_drop mechanism, that is, if the deferred decoding time of $(i+2)^{th}$ frame is larger than one *EDT* after adding the predicted decoding time of the $(i+1)^{th}$ frame, the algorithm checks the frame type of the $(i+1)^{th}$ frame first before the decoding of the $(i+1)^{th}$ frame. If the $(i+1)^{th}$ frame, belongs to *I* or *P* type, then the algorithm allows the decoding of the $(i+1)^{th}$ frame to continue and selects the $(i+2)^{th}$ frame to drop as shown in Fig. 7(c). But if the $(i+1)^{th}$ frame belongs to *B* type, then the algorithm does not allow the decoding of the $(i+1)^{th}$ frame to continue and drop it immediately as shown in Fig. 7(d). There is a decoding idle time between the end of the decoding of the i^{th} frame and the beginning of the decoding of the $(i+2)^{th}$ frame, thus power can be saved. Here we note that Figs. 7(b) and 7(c) share the same diagram essentially.



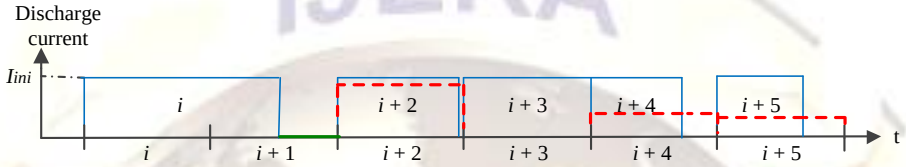
(a) Drop_next mechanism with $(i+2)^{th}$ frame belongs to *I* or *P* type.



(b) Drop_next mechanism with $(i+2)^{th}$ frame belongs to *B* type.



(c) Immediately_drop mechanism with $(i+1)^{th}$ frame belongs to *I* or *P* type.

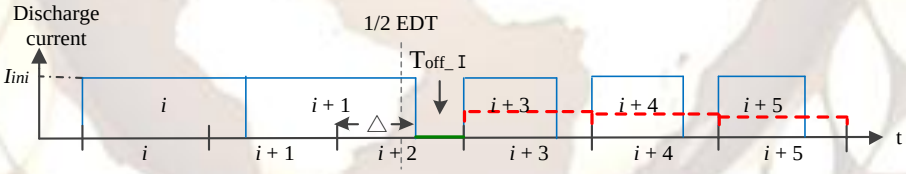


(d) Immediately_drop mechanism with $(i+1)^{th}$ frame belongs to *B* type.

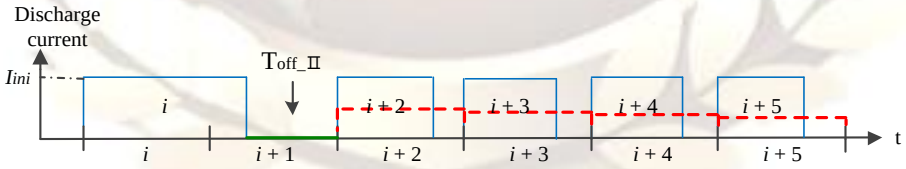
Fig. 7 Schematic illustrations for *Late_dropping* algorithms.

4.2 *Early_dropping* Algorithm

Fig. 8 shows the schematic illustrations for *Early_dropping* algorithm by using immediately_drop mechanism. In Fig. 8(a), if the deferred decoding time of $(i+2)^{th}$ frame is larger than one half of an *EDT* after adding the predicted decoding time of the $(i+1)^{th}$ frame, the algorithm checks the frame type of the $(i+1)^{th}$ frame first before the decoding of the $(i+1)^{th}$ frame. If the $(i+1)^{th}$ frame, belongs to *I* or *P* type, then the algorithm allows the decoding of the $(i+1)^{th}$ frame to continue and selects the $(i+2)^{th}$ frame to drop. Thus, there is a decoding idle time, denoted by T_{off_I}, between the end of the decoding of the $(i+1)^{th}$ frame and the beginning of the decoding of the $(i+3)^{th}$ frame. However, if the $(i+1)^{th}$ frame belongs to *B* type, then the algorithm does not allow the decoding of the $(i+1)^{th}$ frame to continue and drop it immediately as shown in Fig. 8(b). Thus, there is a decoding idle time, denoted by T_{off_II}, between the end of the decoding of the i^{th} frame and the beginning of the decoding of the $(i+2)^{th}$ frame. Although both T_{off_I} and T_{off_II} may save power, it can be easily shown that $T_{off\_II} > T_{off\_I}$.



(a) Immediately_drop mechanism with $(i+1)^{th}$ frame belongs to *I* or *P* type.



(b) Immediately_drop mechanism with $(i+1)^{th}$ frame belongs to *B* type.

Fig. 8 Schematic illustrations for *Early_dropping* algorithm.

5 Evaluation Results

Performance degradation of DVS in MPEG decoding rises due to deadline violations occurred in a burst. We adopt a video clip from a high motion movie "*UnderSiege*" to evaluate the effectiveness of different frame dropping algorithms proposed in the previous section. The maximum decoding frequency for each frame type is set to $f_{I,max}$= 220 MHz, $f_{P,max}$= 160 MHz, and $f_{B,max}$= 90 MHz, where the probability of deadline violations is allowed to about 20%, as we have discussed in section 3. Fig. 9 shows the predicted decoding times of frames numbered from 250 to 700 in the selected video clip. From Fig. 9 we observed that there are two heavy burst intervals (around frames 350-420 and 490-530) and one light burst interval (around frames 600-660). We use these burst intervals to evaluate frame dropping situations for different algorithms. Results are shown in Fig. 10. Briefly, in Fig. 10 we use case I to denote the condition of which *Late_dropping*

algorithm with drop_next mechanism was adopted. Case II denotes the condition of which *Late_dropping* algorithm with immediately_drop mechanism was adopted. Case III denotes the condition of which *Early_dropping* algorithm with immediately_drop mechanism was adopted. Since case I and case II belong to the same *Late_dropping* algorithm, they have similar dropping situations as shown in Fig. 10(a) and (b). On the contrary, case III belongs to *Early_dropping* algorithm, thus has more severe dropping situations as shown in Fig. 10(c). During the first heavy burst interval, case I drops 6 frames and case II drops 7 frames and case III drops 13 frames. During the second heavy burst interval, case I drops 6 frames and case II drops 7 frames and case III drops 9 frames. During the light burst interval, none of the frames were dropped in case I and case II, but case III drops 3 frames. Note that all dropped frames belong to *B* type.

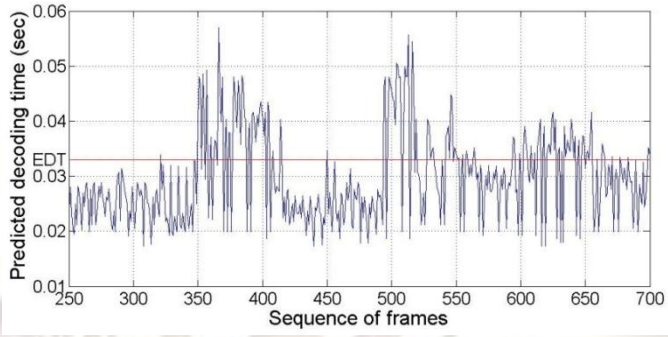
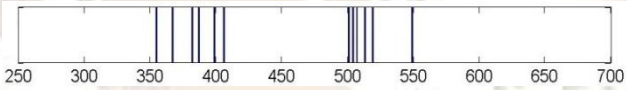
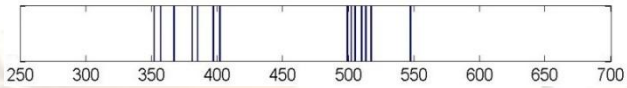


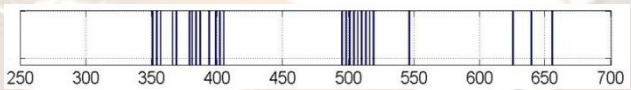
Fig. 9 Predicted decoding time versus frame number in sequence.



(a) Sequence of frame dropping for case I.



(b) Sequence of frame dropping for case II.

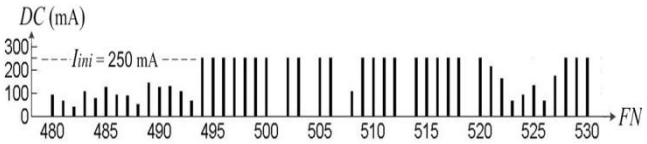


(c) Sequence of frame dropping for case III.

Fig. 10 Sequences of frame dropping for three different cases.

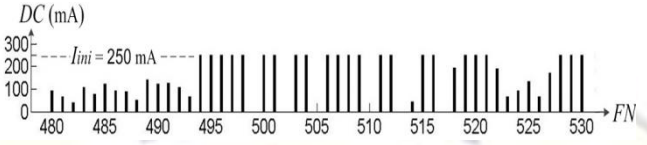
Although case III has almost twice more severe dropping situations than case I and case II, it has better DVS application and thus save more battery energy. Fig. 11 shows the battery discharge current of each frame during the second heavy burst interval for three different cases. We assume all types of frames use the same initial decoding current I_{ini}=250 mA. If DVS cannot apply on the decoding of a frame, its decoding current should maintain the initial value 250 mA. Once the decoding of a frame can adopt DVS, its decoding current can be calculated either by (6) or (7), depending on whether the starting time of the decoding has been deferred. Note that, Fig. 11 shows just the discharge current of each frame, but the actual decoding time of each frame is not shown in the figure. From Fig. 11, we observed that DVS application for three different cases vary from frame 495 to frame 522. During this interval case I drops 5 frames and has 3 times of DVS application as shown in Fig. 11(a). Case II drops 6 frames and also has 3 times of DVS application as shown in Fig. 11(b). Fig. 11(c) shows that case III drops 9 frames and has 6 times of DVS application. We use battery capacity discharge equation (3) to compute the power consumptions for three different cases, from the decoding of frame 480 to frame 530. Fig. 12 shows the battery discharge capacity for three different cases during this interval. Here we observed that before the decoding of frame 495, all three cases have same power consumption begin from frame 480. After that, the power consumptions for three cases differ from each other. Although there have several times that the power consumptions for case I and case II are equal. It still can be seen that case II has better power consumption than case I. The total improvement of power saving during frames 480-530 is about 5.5%. And, it is also obvious that the power consumption for case III is much better than case I and case II. The total improvement of power saving for case III is about 20.8% than case I during frames 480-530. It

is interested to note that for all three cases frame dropping will never be occurred in consecutive. This proofs the effectiveness of the proposed frame dropping algorithms on DVS application in MPEG decoding.

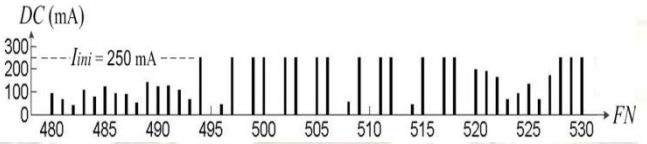


DC : Discharge Current ; *FN* : Frame Number

(a) Battery discharge current for case I.



(b) Battery discharge current for case II.



(c) Battery discharge current for case III.

Fig. 11 Battery discharge current of each frame for three different cases.

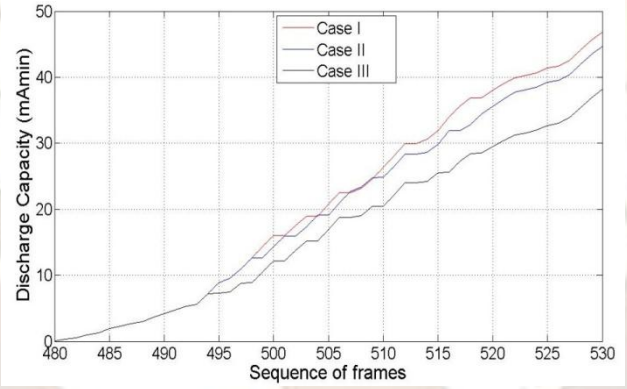


Fig. 12. Battery discharge capacity for three different cases.

6 Conclusions

In MPEG decoding, the decoding time of a frame may be deferred or even the frame may be dropped when frames with large predicted decoding times occurred in a burst. The predicted decoding time of a frame is calcalated by (5), and its value depends on the selected maximum decoding frequency (MDF) for the type of which the frame belongs to. In this paper, we assume the predicted decoding time of each frame can be estimated in advance in an off-line manner. To determine the off-line "suitable maximum" decoding frequency for different frame types, a probabilistic approach is proposed in section 3. To exploit the advantage of DVS, the slack time is reclaimed if the predicted decoding time is smaller than the deadline (*EDT*) to slow down the execution speeds at run time to reduce the energy consumption. The decoding of a frame may be deferred or even the frame may be dropped when frames with large predicted decoding times occurred in a burst. Performance degradation of DVS in MPEG decoding rises due to deadline violations occurred in a burst.

This paper proposes two types of frame dropping algorithms to maintain the energy efficiency of DVS in MPEG decoding; meanwhile, to avoid consecutive frame dropping to preserve the quality of video playing. The *Late_dropping* algorithm selects a frame to drop when the deferred decoding time of a frame is larger than one *EDT*. And the *Early_dropping* algorithm selects a frame to drop when the deferred decoding time of a frame is larger than one half of an *EDT*. We adopt a video clip from a high motion movie "*UnderSiege"* to evaluate the effectiveness of the proposed algorithms. The maximum decoding frequency for each frame type is set to $f_{I,max}$= 220 MHz, $f_{P,max}$= 160 MHz, and $f_{B,max}$= 90 MHz, where the probability of deadline violations is allowed to about 20%. Evaluation results, using a sample heavy burst interval as basis, show that *Early_dropping* algorithm with immediately_drop (case III) mechanism produces

almost twice more dropping situations than that of *Late_dropping* algorithm. But, it acquires better DVS application and thus saves more battery energy to about 20% than that of *Late_dropping* algorithm with drop_next mechanism (case I). *Late_dropping* algorithm with immediately_drop mechanism (case II) produces slightly more dropping situations than that of case I, but it can save more battery energy to 5.5% than that of case I. We also observed that for all three cases frame dropping will never be occurred in consecutive.

More samples of video clips and more samples of time intervals during which large frames occurred in a burst are needed for further performance evaluations of the proposed algorithms. We assume in this paper all types of frames use the same initial decoding current I_{ini}=250 mA. However, this assumption cannot express the real situations. How to determine appropriate initial decoding currents for different types of frames according to the selected maximum decoding frequencies is our future work. The selected maximum decoding frequencies should expolit the advantages of DVS for low-power video decoding; meanwhile, sustain the stability of system performance. Does there has any optimum solution for the selection of those decoding frequencies? This is a hard problem and has many system dependent parameters have to be considered. This study proposes three criteria to drop frames in MPEG decoding. Investigate other algorithms with more efficient dropping criteria also is our future work.

ACKNOWLEDGEMENTS

This work was supported by National Science Council, Taiwan, R.O.C., under the Grant NSC 99-2221-E-224-038.

REFERENCES

[1] D. N. Rakhmatov, S. B. K. Vrudhula, "Energy Management for Battery-Powered Embedded Systems," *ACM Transactions on Embedded Computing Systems*, vol. 2, no. 3, pp. 277-324, Mar. 2003.

[2] W. Chedid, "Dynamic Voltage Scaling Techniques for Power-Efficient MPEG Decoding," Master Thesis, Cleveland State University, Master of Science in Electrical Engineering, 2003.

[3] C. Ma, Z. Zhang, Y. Yang, Y, "Battery-Aware Router Scheduling in Wireless Mesh Networks," *International Parallel and Distributed Processing Symposium*, pp. 228-241, 2006.

[4] A. Rajalingam, B. Kokila, and S. K. Srivatsa, "Low Power MPEG Video Player Using Dynamic Voltage Scaling," *Research Journal of Information Technology*, vol. 1, no. 1, pp. 17-21, 2009.

[5] M. R. Jongerden, B. R. Haverkort, "Which battery model to use?," *IET Software*, vol. 3, no. 6, pp. 445–457, 2010.

[6] R. Rao, S. Vrudhula, and D. N. Rakhmatov, "Battery modeling for energy-aware system design," *Computer*, vol. 36, no. 12, pp. 77-87, 2003.

[7] V. Rao, G. Singhal, A. Kumar, N. Navet, "Battery Model for Embedded Systems," *International Conference on VLSI Design*, pp. 105-110, 2005.

[8] B. Lee, E. Nurvitadhi, R. Dixit, C. Yu, and M. Kim, "Dynamic Voltage Scaling Techniques for Power Efficient Video Decoding," *EUROMICRO Journal*, vol. 51, no. 10-11, pp. 633–652, 2005.