Reducing Multimedia Decode Power using Feedback Control

Zhijian Lu, John Lach, Mircea Stan
Dept. of Electrical and Computer Engineering
University of Virginia
Charlottesville, VA 22904
{zl4j, jlach, mircea}@virginia.edu

Kevin Skadron
Dept. of Computer Science
University of Virginia
Charlottesville, VA 22904
skadron@cs.virginia.edu

Abstract

Despite recent advances, battery life continues to be a limiting factor in mobile multimedia systems. Significant energy savings can be achieved by adapting systems at run-time to match the execution requirements of different tasks. This paper introduces an on-line *dynamic voltage/frequency scaling (DVS) feedback technique that reduces voltage and frequency to match the playback rate. A PI controller adjusts the decoder's speed to keep constant the occupancy of the buffer between the decoder and the display, effectively matching the average decode rate to the display rate without the need for any off-line profiling. MPEG simulation results show that this technique reduces decoder power consumption while providing strong real-time guarantees.*

1. Introduction

Mobile multimedia systems are becoming popular consumer items, but limited battery life continues to be a major problem. Energy-efficiency, however, must be balanced against the fact that users demand a high quality of service (QoS).

This paper considers energy efficiency for multimedia playback, in which a stream of multimedia frames must be decoded before display, and frames that miss their decode deadlines must either be delayed or dropped. Therefore, the frame decode rate must keep up with the defined display rate to avoid choppy playback.

Of course, the decode time for each frame in a multimedia stream is not necessarily uniform. For example, MPEG frames come in three different coding types (intra (I), bi-directional (B), and predictive (P)), each of which requires different decoding effort. Even within these coding types, frame decode time varies. Therefore, a simple constant-rate decoder—running at a speed that enables the worst-case frame to meet its deadline—actually decodes many frames well before their deadlines, creating a great deal of slack that can be reclaimed for energy savings by slowing down the processor. Using dynamic voltage/frequency scaling (DVS), the system can run slower with less power consumption for frames that require less decoding computation. However, this frame information is normally not known before the decode is performed, except at the crude level of whether a frame is of type I/B/P.

This paper introduces an online feedback control technique for DVS in mobile multimedia systems that makes the average frame decode rate the same as the display rate. The key observations are that a buffer between decode and playback provides protection against incorrect DVS settings due to unforeseen changes in decoding complexity, and that the occupancy of this display buffer becomes a natural control object for the system: a draining buffer indicates that the decoding rate is too slow, and a growing buffer indicates that more energy can be saved by scaling down the decoding rate. Although our controller is designed by assuming continuous frequency and voltage scaling, our simulation results show that it works well for discrete frequency/voltage settings.

The advantages of this buffer-based feedback-control approach over prior solutions are that no pre-playback or server-side profiling is required and that slack can be reclaimed across frame boundaries. No explicit frame decode time prediction is needed, thus avoiding the missed deadlines caused by prediction errors. We apply formal feedback control techniques to effectively control the buffer occupancy. The maximum buffer size needed is about 10 decoded frames, which is inexpensive in common hand-held devices [10]. Simulation results demonstrate that our feedback control method approaches the energy savings achievable by the *optimal* scaling method, and up to 20% energy reduction beyond the best on-line technique [6] we are aware of. Moreover, our method provides strong real-time guarantees, for a *zero* deadline miss rate, even without any prior frame information.

The remainder of the paper is organized as follows. Related work is discussed in Section 2. A real-time model of

the multimedia system under study is presented in Section 3 along with analysis of the real-time constraints. Our control algorithm is detailed in Section 4. In Section 5, we present our simulation results and compare our algorithm with different DVS schemes. Finally, we summarize our work in Section 6.

2. Related work

Low power design in multimedia systems is currently a very active research field. The major power saving techniques applied to multimedia systems include DVS and DPM (dynamic power management). In DVS, different computational tasks are run at different voltages and clock frequencies while still providing an adequate level of performance. DPM shuts off system parts (inside or outside the CPU) that are not in use at any given time. Structural adaptations for multimedia systems were recently proposed in [5].

Because a multimedia system has defined QoS requirements, it is usually modeled as a real-time system with soft deadline constraints. Inter- and intra-frame scaling are considered to reduce the deadline miss rate in [5] and [1], respectively. However, in both works, the decoder is assumed to decode only one frame for each display interval. Frame-decode-time prediction techniques are used in these works to achieve a satisfactory tradeoff between the power savings and QoS. Recently, Chung *et al.* [2] proposed that the decoding time information can be obtained from the multimedia content provider, such that the accuracy of the decode time prediction can be improved.

There are two works [6, 10] we are aware of that use buffers in the multimedia system to save power while providing QoS guarantees. Lu *et al.* [10] used an off-line algorithm to schedule the frame decoding rate and respective frequency, and they did not consider multimedia streams that include *B* frames. Im *et al.* [6] focused on the estimation of the input/output buffer size for the decoder, and also proposed an on-line DVS technique. This technique is essentially equivalent to a scheme we call *panic factor scaling* in this paper. We will compare this technique with our feedback control technique.

The design of control systems is a mature field with a history dating back at least as far as the 1600s. Numerous textbooks exist that describe basic control principles, e.g. [3]. Control-theoretic approaches have been applied to a variety of computer system design aspects, including CPU scheduling [9, 16], web server QoS management [7, 11], internet congestion control [4], and data migration [8]. In [12], the authors apply a formal feedback control technique on DVS for multimedia systems, but they focus on average system performance.

3. Background and assumptions

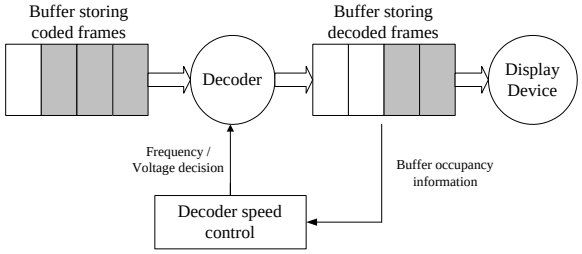

Figure 1. Architecture of the multimedia system studied in this paper.

The general architecture of a multimedia system is illustrated in Figure 1. Frames in the decode buffer (or in disks) are fetched and decoded sequentially and displayed by the display device with a constant playback rate denoted by $\frac{1}{T}$, where T is the display interval. Our technique controls the decode speed such that the energy consumption can be reduced without the frames missing their display deadlines. We make the following assumptions for this multimedia system:

- Coded frames are decoded in their normal order, and a frame is always available to be decoded whenever the frames before it have already been decoded.
- After a frame is decoded and sent to the display buffer, the decoder is ready for decoding the next frame.
- Each frame has an associated deadline, which is the time to start displaying the frame. The deadline is always a multiple of the display interval T. For example, frame i has a deadline of $i * T$. The time needed to decode frame i is denoted by C_i.
- The display buffer has enough space to hold all the frames which are decoded before their deadline.

To clarify our terminology, we state the following simple theorem.

Theorem 1 *Let S be a multimedia stream including frames* $\{F_1, F_2, \ldots, F_n\}$. *If the decoder in the system described above never idles when there are undecoded frames left, S will be schedulable (all the frames meet their respective deadlines) if and only if:*

$$\frac{\sum_{k=1}^{i} C_k}{i} \leq T \qquad \forall i, 1 \leq i \leq n \tag{1}$$

where C_k *is the time needed to decode frame k, and* T *is the display interval.*

The proof of the above theorem is very straightforward. Although we assume that the decoder is dedicated to only

decoding tasks, the theorem can be easily extended to accommodate additional tasks that require the CPU for display (instead of using DMA (direct memory access), etc.) or other real-time tasks (e.g., fetching the coded frames from the disks to the decode buffer). In this scenario, we only need to substitute C_k with the sum of the frame decoding time and the time for additional tasks.

Theorem 1 shows that as long as the average decoding time of the frames (the first frame, the first two frames, the first three frames, etc.) is always less than or equal to the display interval, all the deadlines will be satisfied. This suggests that we can slow down the decoder such that some frames may even have decoding times larger than the display interval, provided that they can make use of the slack time from the frames with shorter decoding times. This gives great flexibility for DVS. In contrast, if a multimedia system has no buffer between the decoder and the display device or the buffer can only hold one frame at a time, the schedulability criteria must be reduced to:

$$C_i \leq T \qquad \forall i, 1 \leq i \leq n \tag{2}$$

which is much more restrictive for DVS.

In this paper, we assume a decoder with DVS and are only concerned with the dynamic energy consumption of the multimedia decoder. We define a variable r, the frequency scaling factor with a value within $[0,1]$, which represents the actual speed at which the decoder will run (each speed (working frequency) is associated with an optimum operating voltage). For example, $r = 1$ means the decoder will run at its full speed, while $r = 0.5$ is half speed. The actual decoding time for frame i can be approximated as $\frac{C_i}{r}$, where C_i is the decoding time at full speed. We adopt the approximate energy calculation from [15] to estimate the dynamic energy consumption of the multimedia decoder:

$$E(r) = r^2 E_0 \tag{3}$$

where E_0 is the energy consumption for a task at full speed, r is the frequency scaling factor, and $E(r)$ is the actual energy consumption at a speed specified by r with the corresponding minimum operating voltage.

4. Control algorithm

In this section, we introduce our DVS feedback control algorithm. The time to switch between DVS settings is assumed to be negligible; some processors stall for as much as 50-100 μsec., but this is less than 1% of the time needed for decoding each frame, and newer processors may reduce or eliminate this latency. In addition, as explained in the following, our control algorithm involves only several add/multiplication operations. The timing and energy overheads are also negligible compared to those for decoding a frame.

4.1. Dead-zone based control algorithm

Before decoding the next frame, we do not know how much time will be required for it. Without a buffer, we must use the most conservative estimate to set the decoding speed. But if there are some decoded frames already in the buffer between the decoder and the display device, we can apply an operating frequency which we predict is close to the optimum. Although the actual decoding time of the frames may vary greatly, its effect on the real-time constraint is hidden by the display buffer. In our algorithm, we always try to control the number of decoded frames in the buffer within a region specified by $[B_l, B_h]$, where B_l is the lower threshold for the number of frames in the buffer and B_h is the higher threshold.

As long as the number of frames in the buffer is within the specified region, we assume that the current decoder speed is the right choice for decoding the frames; i.e. the average decode rate is equal to the display rate. However, if the actual number of frames in the buffer becomes higher or lower than the respective threshold, this means the current decoding speed is too fast or too slow, respectively. We apply a formal feedback controller to pull the number of frames back into the specified region by adjusting the decoding speed.

Since our controller will be applied only when the frame number is out of the specified region, this forms a "dead-zone" based control mechanism. There are two reasons for using a dead-zone: first, we can reduce the overhead of frequent frequency and voltage switches; second, the quadratic energy equation (3) suggests using a constant speed whenever possible in order to minimize the overall energy consumption.

Our frequency scaling factor is composed of two parts:

$$r = r_e + r_c \tag{4}$$

where r_e is the scaling factor estimation based on the decoding information from the previous frames in a certain window size (e.g. 100 frames) and r_c is the outcome from a controller, which can be regarded as an adjustment to r_e in order to pull back the system into the dead-zone. The combinational effect of r_e and r_c will be illustrated in an example scaling curve in Section 5. The calculation of r_e comes from Theorem 1, which provides the schedulability criteria. If we look at the previous frames within a certain window size (labeled as $\{F_1, F_2, \ldots, F_{WindowSize}\}$), a lower bound on the constant decoding speed, r_{lb}, can be obtained from the following equation:

$$\frac{\sum_{k=1}^{WindowSize} \frac{C_k}{r_{lb}}}{WindowSize} = T$$

The non-scaling decoding time C_k can be determined by the scaling factor applied to that frame and the actual tim-

ing measurement for decoding that frame. Although this frequency scaling factor lower bound does not guarantee schedulability for these previous frames, it would give the minimum energy consumption if it did. We use this lower bound as r_e, which will be adjusted with the controller output r_c.

4.2. PI controller

We adopt a formal PI controller [3] to keep the number of frames in the buffer within the dead-zone. This controller is specified by the following equations:

$$r_c = K_p e + K_i \sum e$$

and

$$e = \begin{cases} B_h - b & \text{if} \quad b > B_h; \\ B_l - b & \text{if} \quad b < B_l; \\ 0 & \text{if} \quad B_l \leq b \leq B_h. \end{cases}$$

where e is the control error, which is the difference between the measured control value (i.e., number of frames in the buffer) and the control set-point. e is the input to the controller. K_p and K_i are the proportional and integral gains of the controller, respectively. K_p produces a controller output proportional to the *current* control error, while K_i memorizes the history and produces an output based on the *past* control errors. b is the feedback information about the actual number of frames in the buffer. The controller output, r_c, is used to adjust r_e, as shown in equation (4).

When the system is above the dead-zone ($b > B_h$), the higher threshold is used as the set-point for the controller, and the lower threshold is used when the system is below the dead-zone ($b < B_l$). When the system is within the dead-zone, we let e be 0, just as if the system reaches its set-point, and the controller output r_c is a constant, due to the integral component in the controller. This constant value will be kept effective until the number of frames in the buffer reaches the middle of B_l and B_h. If this happens, it usually implies an over-compensation by the controller, and the controller should be cleared (i.e., $r_c = 0$).

The control problem described in this paper is a difficult problem itself. Unfortunately, we are unable to provide quantitative analysis for the system model as well as the controller design. But as a rule of thumb, smaller controller gain values tend to give stable but less responsive performance, while larger gains tend to make the controller unstable but provide fast response. We found by experiments that small gain values in the controller are sufficient for our purpose.

4.3. Real-time guarantee

Since we control the number of frames in the buffer within a certain range, a frame with an extremely large decoding time will not necessarily result in a missed deadline, but will simply reduce the number of frames in the buffer. However, if the buffer occupancy is too small, a large decoding time may still lead to a missed deadline. One way to solve this problem is for the decoder to detect this situation and jump to a high (safe) speed to decode this long frame. We call the scaling factor corresponding to this high speed the *panic factor*, which we denote by r_p.

If the current number of frames in the buffer is b and the time interval between the present time and the next display time point is Δt, the panic factor can be calculated as follows:

$$r_p = \frac{WCET}{\Delta t + bT} \tag{5}$$

where $WCET$ is the worst-case execution (i.e., decoding) time and $\Delta t + bT$ is the total allowable time to decode the next frame. Panic factor speed is the minimum speed required to avoid a missed deadline. Im *et al.* [6] suggested using this speed to scale the decoder. We will compare this method with our algorithm in Section 5.

In our feedback control algorithm, the final decoding speed will be chosen by:

$$Max\,(r_e + r_c, r_p)$$

The panic factor speed is much more conservative when the display buffer becomes almost empty. This is the best technique we have found so far in the literature to prevent deadline missing. As long as we have accurate knowledge about the worst-case decoding time, we will never miss a deadline with panic factor speed. Of course, it is undesirable to frequently jump to the panic factor, because the panic frequency always tries to spend all available slack time in decoding the next frame. This is very energy-inefficient because the subsequent operating frequency will have to stay high for some period of time. Fortunately, our controller is quite robust and the panic factor is rarely engaged.

5. Simulation experiments and discussion

To demonstrate the benefits of our control algorithm, we compare the performance of different DVS techniques for several different MPEG clips using simulation. We choose MPEG video as our test application because of its large range of variation in decoding time on a frame-by-frame basis. Thus, we believe MPEG video is at least loosely representative of playback of other compressed streams such as MP3 audio. We use *mpeg_play* [13], which is one of the standard utilities to play mpeg-1 streams in UNIX systems, to record timing information for full-speed playback and then use this baseline timing information to simulate the timing and energy performance of several DVS techniques. Decode time for each frame is obtained by scaling

Table 1. MPEG clips used in this paper

MPEG clips	Frame types	Average decoding time (ms)	Worst-case decoding time (ms)	Decoding time standard deviation (ms)	Std./Ave. ratio	Simulated playback rate (Fps)	Total number of frame simulated
vid_small	I,B,P	3.11	10.86	1.62	0.52	35	1500
vid_large	I,B,P	10.36	23.24	3.08	0.30	35	1500
psycho	I,P	14.85	21.64	2.35	0.16	35	1500
dg_2	I,P	9.24	26	3.58	0.39	35	2500
test_motion	I,B,P	15.16	26	1.62	0.11	35	2500
test_nomotion_busybkd	I,B,P	14.27	25.89	2.78	0.20	35	2500
test_nomotion_flatbkd	I,B,P	14.30	26.5	0.69	0.05	35	2500

the full-speed time by the frequency scaling factor, and energy consumption is estimated by applying equation (3).

5.1. Characteristics of the MPEG traces

We tried to choose a set of representative MPEG clips in our simulation experiments. They are listed in Table 1. These clips vary greatly in terms of decoding times, timing variations, and frame types. The two streams *vid_small* and *vid_large* show the same movie but with different quality (160*120 and 320*240, respectively). *Test_motion* is a clip with an abnormally large amount of motion. *Test_nomotion_busybkd* and *test_nomotion_flatbkd* exhibit no motion except during one brief interval near the end of the clip. They differ in that the static scene in *test_nomotion_busybkd* contains many objects and colors, while *test_nomotion_flatbkd* contains a single object against a flat background. Streams *psycho* and *dg_2* have only *I* and *P* frames.

5.2. Comparison of various DVS schemes

We compare the following DVS strategies in our simulator:

• *Ideal_Period*

A decoder with DVS but no display buffer. This will decode only one frame in each display period. In order to meet the deadlines for the frames, condition (2) must be satisfied. The best case is that the decoder will always choose the correct scaling factor such that the actual (scaled) decoding time for each frame is exactly equal to the display period. Of course this scaling method requires that the decoder knows the accurate timing information before decoding that frame. For this strategy, we assume continuous frequency/voltage scaling capabilities. We call this DVS scheme *Ideal_Period*, and it provides the energy consumption lower bound by the DVS techniques in the systems such as those studied in [1, 2].

• *Optimum*

A decoder with a large display buffer and knowledge of the exact timing information for each frame. We developed an off-line scheduling algorithm, which satisfies the schedulability condition (1), while minimizing the total energy consumption. This scheme sets up an achievable energy consumption lower bound by DVS provided that all the deadlines are met. We call this scheme *optimum*. Again, continuous DVS is assumed in this strategy. Because frames can be decoded across display boundaries, a fairly straight scaling curve will be expected in this scheme to achieve the optimal energy savings.

• *Panic factor*

Panic factor scheduling as proposed by Im *et al.* [6], as calculated in equation (5). In order to provide real-time guarantees, accurate worst-case information is assumed to be available. And we apply a frequency/voltage setting which is linearly subdivided into 40 discrete levels from 0 to full speed. This frequency/voltage model is similar to that proposed in [14]. The calculated panic factor will be rounded up to the closest frequency/voltage level.

• *Dead-zone based control algorithm*

Our decoder with a display buffer and our feedback control scheme. As introduced in Section 4, this scheme specifies a region for the number of frames in the buffer. A PI controller is used to pull the system back to this region if the current number of frames in the buffer is out of the specified region. If the lower threshold of this region is too small, deadline missing may occur before the controller can pull the number of frames back up to the region. Our experiments show that $B_l = 3$ is an appropriate choice for this lower threshold. If the width of the specified region is small, the system may go out of the region frequently, causing the scaling factor to oscillate, which is undesirable in terms of energy saving due to the quadratic nature of the energy equation (3). However, too large of a value for the high threshold will increase the cost of the system as well as the energy consumption of the buffer. We chose $B_h = 8$, as it can provide a large cushion space, while limiting the size of the buffer. Due to space limitations, we only report

in this paper the results where the buffer occupancy region is specified by $[B_l = 3, B_h = 8]$, but our simulation results show that larger high thresholds do not necessarily provide better performance gains. Again, a discrete 40-level frequency/voltage setting is used and each frequency factor obtained from our control algorithm is rounded to the closest frequency/voltage level as the final scaling factor decision.

Notice that all of these DVS strategies provide real-time guarantees because the accurate decode timing information or the worst-case information is assumed to be known by these schemes. In our later discussion, this assumption will be relaxed for a more realistic situation. We also considered a *DPM* technique in which one frame is decoded with full speed in each display period , and the decoder will stop until the next period arrives. However, since its energy consumption is usually 3-4 times that of other schemes, we do not present its results in this paper.

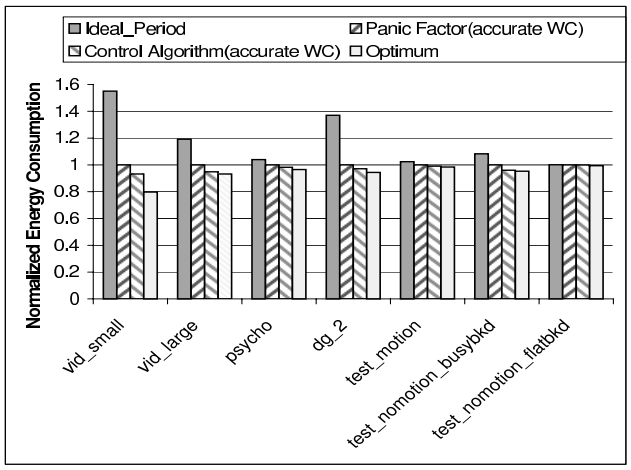

Figure 2. Energy consumption comparison with different DVS schemes. All of the energy is normalized by the energy consumption of the *Panic factor* scheme with accurate worst-case information.

Figure 2 compares the energy consumption under these schemes for different MPEG clips. All energy consumption is normalized by that of *Panic factor* scheme. As one may expect, *Optimum* consumes the least energy because it uses an optimally feasible scheduling method. Another observation is that scaling methods with frame buffering are almost always much better than the scaling scheme for a system decoding one frame per period, because *Ideal_Period* forms the lower bound for energy saving in such systems, e.g., [1, 2]. One exception occurs in the MPEG clip *test_nomotion_flatbkd*, which can be explained by its small timing variation shown in Table 1. When we compare the feedback control algorithm with the panic factor scaling method, our control algorithm always achieves better power savings. Furthermore, our later experiments show that the difference between these two schemes will be enlarged in a more realistic situation, where accurate worst-case decode-timing information is not available. Among all the methods examined, our dead-zone based control algorithm is the closest to the optimal scaling, which gives the lower bound for energy consumption with hard real-time guarantees.

Table 2. Maximum buffer occupancy during the decoding process after the initial frames are drained from the buffer

MPEG clips	Optimum	Panic factor scaling	Feedback control
vid_small	28	7	11
vid_large	13	3	8
psycho	18	2	9
dg_2	24	5	12
test_motion	15	2	10
test_nomotion_busybkd	5	2	9
test_nomotion_flatbkd	5	2	6

Another interesting comparison comes from looking at the buffer occupancy for the scaling methods with frame buffering. Table 2 provides the maximum display-buffer occupancy under the three scaling methods during the decoding process for each MPEG clip. *Optimum* consumes the largest buffer size because it always tries to accumulate as much slack time as possible from the frames with shorter decoding times for the frames with longer decoding times. The maximum buffer occupancy for the feedback control scheme is very close to the specified higher threshold (eight), which proves the effectiveness of our PI controller. The panic factor scheduling method always tries to use up the accumulated slack times, thus the number of frames in the buffer is always small.

5.3. Scaling without accurate worst-case timing information

In the various DVS schemes detailed above, exact or worst-case timing information is used to provide real-time guarantees. However, this information is often unavailable. Moreover, the worst-case decoding time varies significantly among different MPEG clips, as shown in Table 1. Using a uniform worst-case decoding time will be very inefficient in terms of energy savings [2]. A reasonable approach is to predict the worst-case timing information. As an alternative, we can sacrifice the real-time guarantee, which is often acceptable for multimedia applications. However, we will show in the following discussion that our feedback control algorithm provides very robust performance in that it is insensitive to the worst-case decoding time prediction.

Lacking accurate information for an MPEG stream to be displayed, we propose a heuristic technique to estimate the worst-case decoding time. This method is borrowed from the decoding time prediction scheme introduced in [5]. We

use the product of the maximum decoding time currently seen and a prediction factor as the worst-case decoding time estimation. The initial value of the prediction factor is set to 1.1. If the next frame does not miss the deadline, the prediction factor will be reduced by 0.25% until the prediction factor reaches 1.0 or a deadline is missed. If the latter happens, the prediction factor will be reset to its initial value.

Figure 3 illustrates the energy consumption for our control algorithm and for the panic factor method, with accurate/estimated worst-case decoding time. For all the MPEG clips we tested, our control algorithm has almost the same energy consumption regardless of whether the accurate worst-case timing information is used or not. On the other hand, the panic factor scaling method using worst-case prediction consumes more energy than its counterparts with accurate worst-case information for those MPEG clips with large timing variation such as *vid_small* and *vid_large*.

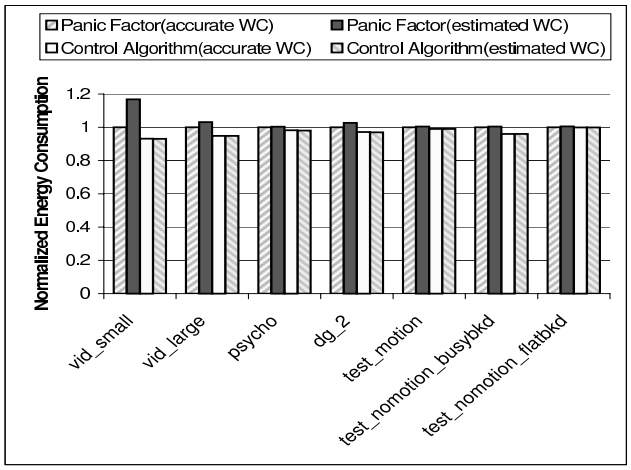

Figure 3. Energy consumption comparison for the panic factor scaling method and the feedback control approach with/without accurate worst-case decoding time information. All of the energy is normalized by that of the *Panic factor* scheme with accurate worst-case information.

As mentioned above, when worst-case estimation is used, hard deadline guarantees cannot be made in theory. Table 3 lists the number of frames missing their deadlines for both methods, as well as the energy consumption. While the panic factor scaling always produces missed deadlines due to the inaccurate worst-case information, our control algorithm still provides strong real-time performance. Except for one clip (*vid_small*), our feedback control approach consumes no more than 3% more energy than *optimal* scaling, while it consumes up to 20% less energy than the *panic factor scaling* method.

As an example, the frequency scaling process for one clip (*dg_2*) is shown in Figure 4 including three different scaling approaches: *optimum*, *feedback control*, and *panic factor scaling*. In the optimum method, the scaling curve is extremely smooth, as we can expect from the energy equation (3). At the other extreme, the panic factor method produces a very noisy scaling factor, due to its rather ad hoc nature. Our feedback control method produces a scaling curve in between, suggesting possible energy saving compared with the panic factor scaling method. Also seen from Figure 4(b) is that our controller is very effective to pull back the system into the specified "dead-zone". When the system is within the "dead-zone", the scaling factor is mainly determined by r_e (see equation (4)), which is rather smooth (e.g., the scaling segment after the 500th frame). Whenever the frame number in the buffer is outside the "dead-zone", the controller is engaged, and, as an adjustment to r_e, the controller output r_c is changed rapidly in the opposite direction of the change in the frame number until the system returns to the "dead zone" again. For example, the scaling segment before the 1500th frame illustrates such a returning process. The two spikes shown in the scaling curve are due to our conserving real-time guarantee scheme as discussed in Section 4.3.

Table 3. Performance comparison between panic factor scaling and our feedback control approach without accurate worst-case information

MPEG clips	Panic factor scaling		Feedback control		Energy saving by feedback control over panic factor
	Frames missing deadline	Energy more than *Optimum*	Frames missing deadline	Energy more than *Optimum*	
vid_small	5	46.5%	0	16.7%	20%
vid_large	6	10.7%	0	1.8%	8%
psycho	1	3.9%	0	1.5%	2.3%
dg_2	4	8.8%	0	2.8%	5.5%
test_motion	4	2.0%	0	0.6%	1.4%
test_nomotion_busybkd	4	5.4%	0	0.7%	4.4%
test_nomotion_flatbkd	3	1.1%	0	0.5%	0.6%

6. Conclusions and future work

This paper introduced a control feedback approach to DVS for reducing power consumption in multimedia decoders. A PI controller is used to keep the decoder rate as close as possible to the display rate without any prior knowledge of the multimedia stream or individual frames. When compared to other DVS approaches, the control feedback technique enables the power consumption to approach that of the optimal scaling method, while providing strong real-time guarantees even without worst-case decode information.

Future work will include investigating the power consumption of the display buffer, although the maximum buffer size required for the control feedback approach is

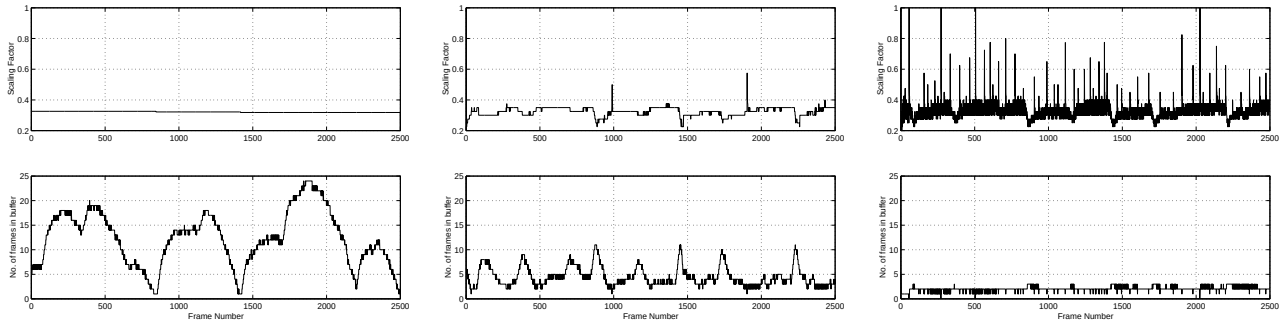

Figure 4. Scaling process for 3 different DVS schemes. (a) Optimal scaling. (b) Feedback control algorithm. (c) Panic factor method. The number of decoded frames in the buffer is also shown along the decoding process for each DVS scheme.

minimal. In addition, we will explore algorithms that can handle non-continuous input streams (e.g., networked streams), in which a frame may not always be available for decoding. We will also extend our research to systems with multiple real-time tasks. Finally, this technique will be implemented in a real system for measured experiments.

7. Acknowledgments

This work is supported in part by the National Science Foundation under grant Nos. CCR-0105626, CCR-0133634, and a grant from Intel MRL. We would also like to thank the anonymous reviewers for their helpful comments.

References

[1] K. Choi, K. Dantu, W. Cheng, and M. Pedram. Frame-based dynamic voltage and frequency scaling for a MPEG decoder. In *Proceedings of International Conference on Computer Aided Design*, pages 732–37, November 2002.

[2] E. Chung, L. Benini, and G. D. Micheli. Contents provider-assisted dynamic voltage scaling for low energy multimedia applications. In *International Symposium on Low Power Electronics and Design*, pages 42–47, August 2002.

[3] G. F. Franklin, J. D. Powell, and M. L. Workman. *Digital Control of Dynamic Systems*. Addison-Wesley, third edition, 1998.

[4] C. V. Hollot, V. Misra, D. Towsley, and W. Gong. A control theoretic analysis of RED. In *Proceedings of IEEE INFOCOM*, April 2001.

[5] C. J. Hughes, J. Srinivasan, and S. V. Adve. Saving energy with architectural and frequency adaptations for multimedia applications. In *Proceedings of the 34th Annual International Symposium on Microarchitecture*, December 2001.

[6] C. Im, H. Kim, and S. Ha. Dynamic voltage scheduling technique for low-power multimedia applications using buffers. In *International Symposium on Low Power Electronics and Design*, pages 34–39, August 2001.

[7] C. Lu, T. F. Abdelzaher, J. A. Stankovic, and S. H. Son. A feedback control approach for guaranteeing relative delays in web servers. In *Proceedings of the IEEE Real-Time Technology and Applications Symposium*, June 2001.

[8] C. Lu, G. A. Alvarez, and J. Wilkes. Aqueduct: Online data migration with performance guarantees. In *Proceedings of the USENIX Conference on File and Storage Technologies*, pages 219–230, January 2002.

[9] C. Lu, J. A. Stankovic, G. Tao, and S. H. Son. Feedback control real-time scheduling: Framework, modeling, and algorithms. *Real-Time Systems Journal*, 23:85–126, 2002.

[10] Y. Lu, L. Benini, and G. D. Micheli. Dynamic frequency scaling with buffer insertion for mixed workloads. *IEEE Transactions on computer-aided design of integrated circuits and systems*, 21(11):1284–1305, November 2002.

[11] Y. Lu, A. Saxena, and T. F. Abdelzaher. Differentiated caching services: a control-theoretical approach. In *Proceedings of the International Conference on Distributed Computing Systems*, April 2001.

[12] Z. Lu, J. Hein, M. Humphrey, M. Stan, J. Lach, and K. Skadron. Control-theoretic dynamic frequency and voltage scaling for multimedia workloads. In *Proceedings of the 2002 International Conferences on Compilers, Architecture, and Synthesis for Embedded Systems*, pages 156–163, October 2002.

[13] L. A. Rowe, K. Patel, and B. C. Smith. Performance of a software MPEG video decoder. In *Proceedings of ACM Multimedia*, August 1993.

[14] G. Semeraro, G. Magklis, R. Balasubramonian, D. H. Albonesi, S. Dwarkadas, and M. L. Scott. Energy-effi cient processor design using multiple clock domains with dynamic voltage and frequency scaling. In *Proceedings of 8th International Symposium on High-Performance Computer Architecture (HPCA'02)*, February 2002.

[15] A. Sinha and A. P. Chandrakasan. Energy effi cient real-time scheduling. In *Proceedings of the International Conference on Computer Aided Design (ICCAD)*, November 2001.

[16] D. C. Steere, A. Goel, J. Gruenburg, D. McNamee, C. Pu, and J. Walpole. A feedback-driven proportion allocator for real-rate scheduling. In *Proceedings of the Third Symposium on Operating System Design and Implementation*, pages 145–158, February 1999.