Design and Implementation of an Energy Efficient Multimedia Playback System*

Zhijian Lu†, John Lach†, Mircea Stan†, Kevin Skadron‡,
Departments of †Electrical and Computer Engineering and ‡Computer Science, University of Virginia
Charlottesville, VA 22904
{zl4j, jlach, mircea}@virginia.edu, skadron@cs.virginia.edu

Abstract

Mobile devices capable of multimedia playback have become popular consumer items, making techniques for energy management during multimedia decoding increasingly important. In this paper, we model the multimedia decoding process as a discrete-time system excited by random input sequences representing the incoming stream and therefore, unlike many existing techniques, do not make any assumptions about the workload streams. Using this novel stochastic process model, we can apply formal methods to analyze the decoding system and design a feedback-based on-line dynamic voltage/frequency scaling (DVS) algorithm to effectively reduce the energy consumption during multimedia playback. We implemented our technique in a laptop computer equipped with a DVS enabled processor, and results reveal good performance for real-world video clips across a wide range of video compression formats compared with existing techniques.

1. Introduction

While the continuously increasing computation power provided by technology scaling enables more complex mobile applications, energy consumption becomes a limiting factor. Since multimedia playback is among the dominant applications in mobile devices, it is important to design energy efficient multimedia playback systems. Fortunately, due to the variations in multimedia decoding complexity, the required computation power changes during the playback, and dynamic voltage/frequency scaling (DVS) is a powerful technique to exploit this opportunity to reduce runtime power by lowering down the circuit speed (and thereby power) whenever possible. However, DVS multimedia systems should be carefully designed to avoid causing large degradation in playback quality. In this paper, we systematically analyze the design of such a system, propose a new design technique and validate our design through a prototype system on a DVS enabled laptop computer.

*This work is supported in part by the National Science Foundation under grant Nos. EHS-0410526, CCF-0429765, CCR-0133634 (Career) and EHS-0509245.

Designing a good multimedia playback system using DVS involves trade-offs among multiple contradicting objects and is subject to practical constraints. Techniques proposed in [5, 6, 10] require perfect predictions for decode execution timing, which is not possible in some multimedia compression formats. The methods in [7, 8] exploit buffering to improve the energy efficiency of DVS. However, smaller buffer sizes might increase the frame deadline miss rate, while larger buffer sizes not only increase hardware costs but also increase the playback latency, which is unacceptable in many real-time applications. In general, a practical solution has to seek the best trade-off between power consumption, playback quality and hardware resources and, in general, has to assume no specific information about the incoming multimedia streams.

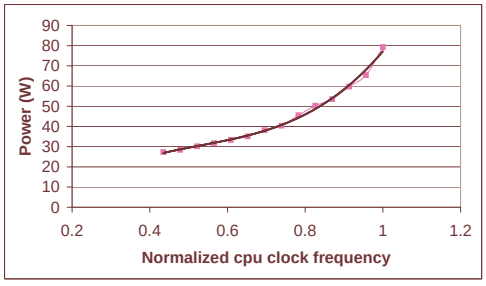
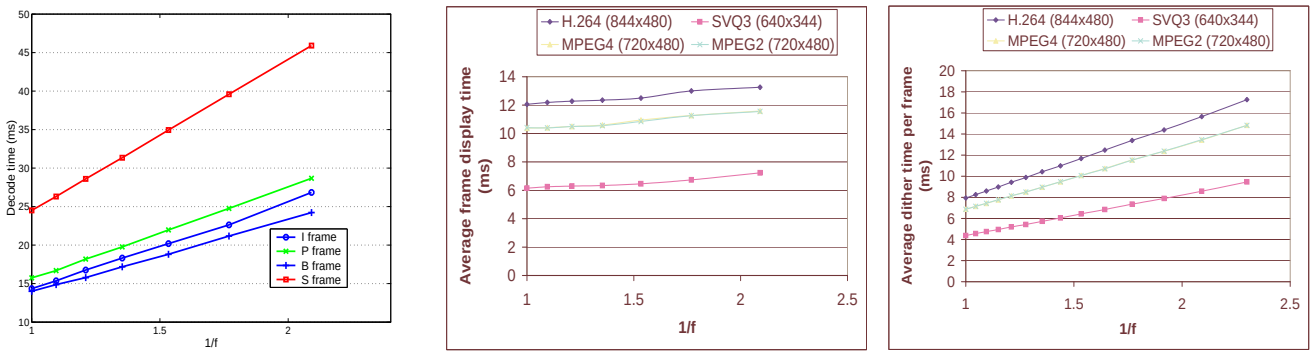


Figure 1. MPEG decoding power at different CPU speed.

The DVS design presented in this paper provides a good balance between all these competing factors. We first create a model for the available design space. Then we search the design space with two desired properties in mind: speed schedule uniformity and system stability. Power consumption is a convex function of circuit speed. As an example, in Figure 1, the MPEG decode power consumption of a laptop computer is shown at different CPU speeds (i.e., frequency and voltage settings). Though each frame can be decoded at a different speed due to computation variations, decoding multiple frames at a uniform speed (i.e., the average decoding speed) provides better energy efficiency according to Jensen's inequality [9]. Therefore, in an energy efficient



(a) (b) (c)

Figure 2. Timing characteristics of MPEG video playback.

DVS design, a uniform speed schedule is preferable. On the other hand, due to the complexity in many modern multimedia coding formats, a good DVS solution should not assume any prior knowledge or rely on any accurate predictions about the decoding process. Therefore, the decode speed decisions should be solely dependent on the results of the previous frames. Any system working in this way is in fact a closed loop feedback system for which stability is an important property.

The rest of the paper is organized as follows. In Section 2, we present some video playback timing characteristics from actual measurements of video clips with various compression formats. Based on these observed timing characteristics, in Section 3, we first propose a generic architecture to model the dynamics of a variable speed video decoding system. Then we discuss the design of the speed controller based on speed uniformity and system stability, and we propose an enhanced feedback based speed control scheme. In Section 4, we describe our implementation of various DVS techniques for video playback on a DVS enabled platform. Finally, Section 5 presents the performance comparison of different DVS techniques on a set of clips with various video formats and shows the effectiveness of the new technique proposed in this paper.

2. Timing characteristics of MPEG workloads

In an MPEG playback application, there are two major tasks: frame decoding and frame display. Though different video compression formats have quite different implementation details, there are several common steps in frame decoding: a. Huffman decoding, b. IDCT and motion estimation/compensation, and c. frame dithering (i.e., converting the decoded frame from the YUV plane to the RGB plane). The decoded frame in RGB format is then sent to a graphics device that finishes the display task. There are several frame types depending on the operations needed to encode/decode a frame: I (intra-coded) frame, P (predictive coded) frame, B (bidirectionally predictive-coded) frame and S (sprite) frame (coded using global motion estimation) in MPEG4.

Figure 2(a) plots the decoding time (including dithering) of different frame types versus the inverse of decode speed (i.e., CPU clock frequency). This figure indicates that the decoding time of a frame is strongly dependent of the decoding speed and can be modeled as $t = \frac{k}{u} + m$, where t is the decode time, u is the CPU speed, and k and m are model parameters representing computation time and memory access time, respectively, during decoding. k is usually several times larger than m, which implies that MPEG decoding is a computation-dominated application.

On the other hand, frame display time is almost insensitive to CPU speed as illustrated by Figure 2(b). This suggests that energy saving opportunities are available during frame display since one can lower the CPU frequency and voltage when the frame is sent to the screen without affecting playback quality. Choi*et al.* [6] made a similar observation in their study. They also reported that the time spent in frame dithering is also insensitive to CPU speed. We plotted the frame dithering time at different CPU speeds in Figure 2(c) and found that the dithering time in the video clips we tested is actually strongly dependent on CPU speed.

3. Design and analysis of feedback based speed control

3.1. Model of DVS multimedia decode systems

Most existing online DVS decoding schemes [5, 6, 7, 8] determine the required decoding speed for future frames according to the decode timing information of previous frames. In this paper, we first propose a dynamic system model, shown in Figure 3, to represent a design space containing many of these existing schemes.

In order to explain this model more clearly, we introduce the concept of frame slack time, defined as the time interval between when the frame decoding begins and the frame deadline (the time to display the frame). The slack time for frame n is the maximum available time for its de-

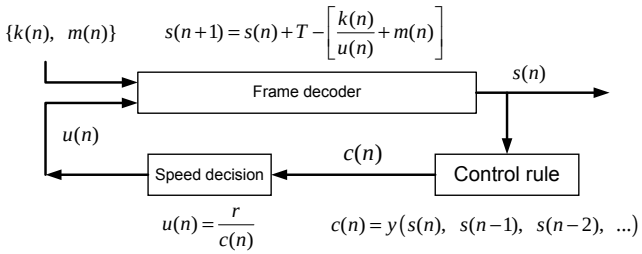


Figure 3. A generic dynamic system model for online DVS MPEG decoding.

coding, denoted by $s(n)$. As discussed in Section 2, the decoding time of a frame can be expressed as $\frac{k(n)}{u(n)} + m(n)$. Let T represent the frame display interval during playback, and we can update the slack time for the next frame by $s(n+1) = s(n)+T-[\frac{k(n)}{u(n)}+m(n)]$, as represented by the "Frame decoder" block in Figure 3. By modeling the decoding time for a frame as a random variable, an MPEG stream can be represented by a sequence composed of random variables $k(n)$ and $m(n)$. Therefore, the MPEG decoding process can be modeled as a discrete time system excited by a stochastic input sequence as shown in Figure 3. The feedback path in Figure 3 represents a generic approach to decode future frames based on previous frames, as used in many existing online DVS MPEG decoding schemes. This generic model makes it possible for us to explore the design space more systematically.

The major difference in various online schemes lies in the parameter r and the control function $y()$. For example, in the frame based schemes [5, 6], $c(n) = \frac{s(n)-m'(n)}{k'(n)}$ and $r = 1$, where $k'(n)$ and $m'(n)$ are the predictions for $k(n)$ and $m(n)$ of frame n, based on previous decoding results of $s(n), s(n-1), \ldots$. In the scheme proposed by Im and Ha [7], $r = WCET$ and $c(n) = s(n)$ where $WCET$ denotes the worst-case frame decoding time. When a formal PI controller is adopted as the control rule as the one proposed by Lu *et al.* [8], the control function $y()$ has the form: $c(n) = \frac{1}{k_p(s(n)-s_0)+k_i\sum_{i=1}^{n} s(i)-s_0}$ and $r = 1$, where k_p and k_i are the proportional and integral gains, respectively, and s_0 is the setpoint of the controller input.

3.2. Design of an enhanced feedback control scheme with speed uniformity

Since uniform speed provides better energy savings when the decoding throughput is matched with that of display, speed uniformity is an ideal property for any DVS scheme. Keeping this in mind and using the dynamic model shown in Figure 3, it is possible to design a better feedback based speed control scheme for MPEG decoding.

Assuming that the system described in Figure 3 is operated under a constant speed, the expected value of random variable $s(n)$ (i.e., the frame slack time), denoted by $S(n)$, satisfies the following relation:

$$S(n+1) = E[s(n+1)] = E\left[s(n) + T - [\tfrac{k(n)}{u(n)} + m(n)]\right] = E[s(n)] = S(n)$$

since $E\left[T - [\frac{k(n)}{u(n)} + m(n)]\right] = 0$, which is required by the system steady state condition that the decoding throughput is matched with the display throughput. Therefore, in the desired DVS schedule, both the decoding speed and the average value of frame slack time are constant. Since the frame slack time is observable in the system, we propose to use the average slack time to determine the decoding speed, i.e., a control rule $c(n) = y(S(n))$ and $r = 1$.

Using this control rule, the feedback system can be described by a difference equation:

$$s(n+1) = s(n) + [T - [y(S(n))k(n) + m(n)]] \quad (1)$$

Assuming that $\{k(n), m(n)\}$ are IID (independent, identical distribution) random sequences[1] and $E[k(n)] = K, E[m(n)] = M$, one can apply the expected value function on both sides of Equation (1).

$$S(n+1) = S(n) + [T - [y(S(n))K + M]]$$

In the steady state, $S(n+1) = S(n) = S$ and $T = y(S)K + M$, and it follows that the steady decoding speed is

$$U = \frac{1}{y(S)} = \frac{K}{T-M} \quad (2)$$

The expected value of $s(n)$ can only be estimated from the decoding results $\{s(n), s(n-1), \ldots\}$. We adopt a simple moving average function, $s'(n) = \frac{s(n)+s(n-1+\ldots+s(n-i+1))}{i}$ as the estimation of E[s(n)], in which i is the window width of the moving operation and $s'(n)$ is also a random variable. During runtime, even if the decoding speed is correctly chosen for frame n (i.e., $u(n) = \frac{K}{T-M}$), $u(n+1)$ could be different due to the difference between $s'(n+1)$ and $s'(n)$. The variation of $s'(n)$ can be estimated using the standard deviation of $\Delta s'(n) = s'(n) - s'(n-1)$. Let $\sigma()$ denote the standard deviation function, and it can be shown that:

$$\begin{aligned}\sigma(\Delta s'(n)) &= \tfrac{1}{\sqrt{i}}\sigma(T - [\tfrac{k(n)}{U} + m(n)]) \\ &= \tfrac{1}{\sqrt{i}}\sqrt{\tfrac{\sigma^2(k(n))}{U^2} + \sigma^2(m(n))} \approx \tfrac{\sigma(k(n))}{\sqrt{i}}\tfrac{1}{U}\end{aligned} \quad (3)$$

because $\sigma(k(n))$ is usually much larger than $\sigma(m(n))$, as found from the timing measurements of some MPEG clips

[1] Frames in a clip are decoded in a predefined order specified by GOP (Group of Pictures). Therefore two consecutive frames are not necessarily statistically independent. The IID assumption only serves to reduce the analysis complexity.

in Section 2. Therefore, the variation in speed decision $\Delta u(n) = u(n) - u(n-1)$ due to the variations in frame decoding time can be estimated according to $u(n) = \frac{1}{y(s'(n))}$:

$$\sigma\left(\Delta u(n)\right) \approx \frac{-y'(s'(n))}{y^2(s'(n)}\sigma\left(\Delta s'(n)\right) \tag{4}$$

If we can assume that the variations of the amount of computation in frame decoding is proportional to the average amount of computation, we have $\sigma(k(n)) \propto K$. It follows that $\sigma(\Delta s'(n))$ is independent of $s'(n)$ (i.e., current speed) because of Equation (2) and (3). We would like to choose a function $y(x)$ such that $\sigma\left(\Delta u(n)\right)$ is also independent of the current decoding speed or $s'(n)$. According to Equation (4), this requires that:

$$\frac{y'(x)}{y^2(x)} = C$$

where C is a constant. Thus, the control function $y()$ in Figure 3 has the form $c(n) = y(s'(n)) = \frac{1}{as'(n)+b}$, where a and b are two constant parameters to be determined. In other words, we have derived a simple linear control rule to determine the next frame decoding speed as

$$\begin{array}{c} c(n) = y(s'(n)) = \frac{1}{as'(n)+b} \\ \text{and} \\ u(n) = \frac{1}{c(n)} = as'(n) + b. \end{array} \tag{5}$$

Essentially, this is a P (proportional) controller.

The parameters a and b can be determined according to practical constraints such as available buffer size and decoding speed (clock frequency) range. For example, we can conservatively set the full speed when the frame slack time is less than the display period and set the minimum speed when the display buffer is full. Let B denote the number of available display buffer sizes, we have $Max(s'(n)) = (B+1) * T$. These settings require that:

$$\left\{ \begin{array}{l} a * T + b = u_{max}; \\ a * [(B+1)T] + b = u_{min}. \end{array} \right. \tag{6}$$

where u_{max} and u_{min} are the maximum and minimum decoding speeds provided by the system. The values of a and b can be determined accordingly and obviously $a < 0$.

3.3. Stability analysis of the proposed feedback control system

In the above analysis, we assume the input sequences $\{k(n), m(n)\}$ are stationary (i.e., K and M are independent of time). During movie playback, we find that some clips show strong phased behaviors. In order to provide both good playback quality and low energy consumption,

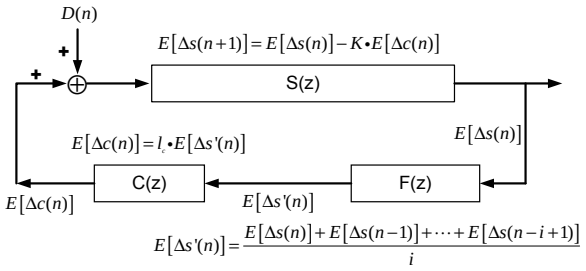


Figure 4. Modeling the transient behaviors of the proposed closed loop feedback system. A pulse input $D(n)$ models the disturbance caused by the change in the required decode computation power.

the feedback system should be able to adapt the speed decision to the new set of K and M. This requires the system to have a fast response time and be stable.

The proposed feedback control system can be analyzed using well established linear system analysis techniques (e.g., transfer functions). In order to do so, we first linearize the proposed control function (Equation (5)) using the first order Taylor expansion:

$$\begin{array}{c} c(n) \approx c(n-1) + l_c\left[s'(n) - s'(n-1)\right] \\ \text{and} \\ l_c = y'(E\left[s'(n-1)\right]) = \frac{-a}{E^2[u(n-1)]}. \end{array} \tag{7}$$

Second, instead of directly analyzing the signals shown in Figure 3, we focus on the signal difference at consecutive time steps, e.g., $\Delta s(n) = s(n) - s(n-1)$, $\Delta c(n) = c(n) - c(n-1)$, etc. In the steady state, the decoding rate matches the display rate, and $E[\Delta s(n)] = 0$. When the required computation power is changed during the playback (i.e., the values of K ($E[k(n)]$) and M ($E[m(n)]$) change), $E[\Delta s(n)]$ will become non-zero and break the steady state. If the system is stable, $E[\Delta s(n)]$ will converge to zero again and the system reaches a new steady state. The block diagram in Figure 4 models this dynamic process. The changes in K and M are modeled by injecting a disturbance signal $D(n)$ (usually a pulse signal) into the system, as shown in Figure 4.

One important difference between Figure 3 and Figure 4 is that the sequences in Figure 4 are formed by applying the expected value function to those stochastic sequences in Figure 3, because only expected values of those random variables are amiable for analysis using transfer functions. Other differences between these two figures include: 1. the blocks "Frame decoder" and "speed decision" in Figure 3 are combined into one block denoted by $S(z)$, and 2. one more block is added to model the moving average function introduced in the proposed feedback system.

It can be verified that the transfer functions in Figure 4 are: $S(z) = \frac{K}{1-z}$, $C(z) = l_c$, and $F(z) =$

$\frac{1}{i}\frac{z^{i-1}+z^{i-2}+\ldots+1}{z^{i-1}}$. Consequently, the relation between $D(n)$ and $E[\Delta s(n)]$ can be created using their $\mathcal{Z}$ transforms.

$$\frac{\mathcal{Z}\{E[\Delta s(n)]\}}{\mathcal{Z}\{D(n)\}} = \frac{S(z)}{1-S(z)C(z)F(z)} = \frac{-Kz^{i-1}}{z^i-(1-\frac{Kl_c}{i})z^{i-1}+\frac{Kl_c}{i}(z^{i-2}+z^{i-3}+\ldots+1)}$$

Therefore the characteristic equation for the closed loop system in Figure 4 is

$$z^i-(1-\frac{Kl_c}{i})z^{i-1}+\frac{Kl_c}{i}(z^{i-2}+z^{i-3}+\ldots+1)=0 \quad (8)$$

The system is stable if and only if the roots of Equation (8) are within the unit circle. The magnitude of the roots determines the converging speed (i.e., response time) of the system.

Figure 5 shows the root with the maximal magnitude of Equation (8) at different values of Kl_c and i. As indicated by the figure, as the moving averaging window size (i) increases, the system tends to be unstable and reacts more slowly. This is expected as the new feedback signals are weighted less with larger window sizes.

From (7), we have $Kl_c \leq K\frac{-a}{u_{min}^2} \leq \frac{-aT}{u_{min}^2} = \frac{u_{max}-u_{min}}{Bu_{min}^2}$, because of $a=-\frac{u_{max}-u_{min}}{BT}$ from (6) and assuming $K \leq T$. Thus, the stability condition for the system under study can be written as:

$$\frac{u_{max}-u_{min}}{Bu_{min}^2} < P_i \quad (9)$$

where there are roots on the unit circle when $Kl_c = P_i$ in Equation (8).

Strictly speaking, condition (9) only guarantees the stability of the linearized closed loop system shown in Figure 4 and, consequently, the stability of the original system when the system is operating near the steady state. However, condition (9) does not necessitate the global stability of the original system due to the non-linear control function (Equation (5)). The study of the global stability of the closed loop non-linear system is outside the scope of this paper. In practice, we found that condition (9) is sufficient to ensure stability in our experimental system.

3.4. Real-time guarantee

High video playback quality is accomplished by ensuring that frames are decoded before their deadlines. In a feedback based DVS decoding system (e.g., the system shown in Figure 3), real-time guarantees are achieved by the control function $c(n)$. Specifically, for the proposed control function in Equation (5), we find that when $i=1$, no frames will miss their deadlines if the worst-case decoding time at full speed is less than the display interval T and:

$$B \geq \frac{u_{max}-u_{min}}{u_{min}} \quad (10)$$

in which B is the display buffer size.

When $i>1$, condition (10) no longer provides hard real-time guarantee. But it still provides helpful reference for choosing the correct design parameter.

3.5. Trade-off analysis on design parameters

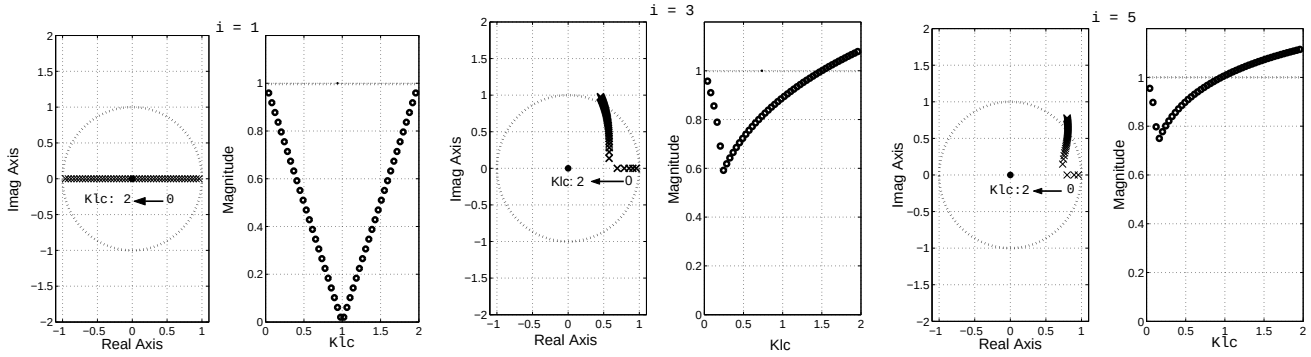
For a practical DVS system providing variable speed between u_{max} and u_{min} and a target display rate (or display interval T), there are two design parameters for the proposed feedback based decoding system: display buffer size B and moving average window size i. According to Equations (4) and (5), the decoding speed variation can be rewritten as $\sigma(\Delta u(n)) \approx a\sigma(\Delta s'(n))$ where a is the control function parameter in Equation (5) and $\sigma(\Delta s'(n))$ is defined in Equation (3). Equations (6) and (3) indicate that larger B and i result in smaller a and $\sigma(\Delta s'(n))$, respectively, and therefore smaller speed variation, which is preferable for energy savings. In addition, larger display buffer sizes help satisfy the stability condition (9) and reduce frame deadline misses as implied by Inequality (10).

On the other hand, there are disadvantages for larger values of B and i. Larger display buffer sizes not only increase hardware costs, but they also increase the playback latency because $B+1$ frames are decoded ahead of displaying in the worst-case. This is especially undesirable in some real-time multimedia applications. The disadvantages of larger values of i can be seen from Figure 5. A larger i tends to increase the magnitude of the roots of the characteristic equation of the feedback system, thereby increasing the system response time. It also reduces the stability range of Kl_c, causing the system to be unstable.

4. Implementation on a hardware platform

Our prototype platform is a Compaq notebook computer equipped with a mobile AMD Athlon XP DVS enabled processor. The CPU clock frequency can be dynamically scaled to 14 discrete speeds between 665MHz and 1530MHz by writing the voltage/frequency setting to a model specific register (MSR). During the experiments, we pull out the notebook battery and put a small sense resistance ($10m\Omega$) in series with the laptop. The voltage drop on the sense resistance is amplified and sampled at a $5K$ sample rate by a desktop computer equipped with a data acquisition card and LabVIEW software.

The video playback software is implemented using two processes and runs under the Linux operating system. One process of the software is responsible for fetching coded frames from the hard disk, decoding the frame and putting the decoded frame in the display buffer. The other process is responsible for displaying frames on the screen by fetching a frame picture from the buffer and sending it to the X server. The display process is in sleep mode most of time and is periodically woken up by a timer that repeatedly



(a) (b) (c)

Figure 5. Roots with the maximal magnitude as functions of Kl_c. (a) $i = 1$. (b) $i = 3$. (c) $i = 5$. The plots on the left show the locations of the roots in the unit circle and those on the right show the magnitude of the roots.

times out at the display interval (i.e., the inverse of the playback rate). Frame deadline misses occur when the display process tries to fetch a frame picture from an empty buffer. When the display process is woken up, the decode process is suspended and resumed after the display process is finished. At the beginning of decoding a new frame, the decode process determines the decoding speed according to the specific DVS scheme and changes the CPU frequency/voltage using a system call. The open source codec library *ffmpeg* [1] is used in the decode process. Thus, video clips with a wide range of different formats can be played in this software.

5. Experimental results

We implemented various DVS MPEG decoding schemes in our hardware platform and tested them on a set of eight video clips with different compression formats, including MPEG2/4, H.264 and SVQ (QuickTime movie format). Among the clips, six are movie trailers downloaded from the Internet [2, 3, 4]. The other two, (*fs2003* and *fs2004*), are home-made movies. The various DVS schemes implemented are similar to those examined in [8] plus the new feedback scheme introduced in this paper:

• *Full speed* The CPU always decodes frames at full speed just as in the systems without DVS capabilities. The CPU becomes idle until the decoded frame is displayed. Therefore the display buffer size in the scheme is 1.

• *Ideal period* Using profiling information for the video streams, the processor calculates the correct decoding speed such that the decoding time for each frame is exactly equal to the display period and rounds up this calculated speed to the available discrete speed provided by the hardware. This scheme is equivalent to the one in [6] when their predictions about frame decode timing are always accurate.

• *Optimum* In this scheme, an off-line scheduling algorithm is developed to find the best schedule such that no frame will miss its deadline while total energy consumption is minimized. The profiled timing information for all frames has to be used to find the optimal schedule, and this scheme sets up an achievable energy consumption lower bound with DVS. This scheme was first proposed in [8] and later revisited in [10].

• *Panic* This scheme is similar to that proposed by Im *et al.* [7], which tries to spend the available slack time in decoding the next frame and assumes the decoding time of the next frame is equal to $WCET$. Again the calculated speed is rounded up to the next available discrete speed. The display buffer size for this scheme is 5.

• *Dead-zone based feedback PI controller* This scheme specifies a region (dead-zone) for the number of decoded 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 dead-zone. Following the design in [8], in our implementation, the dead-zone is between 3 and 8 frames. When the system is within the dead-zone, the CPU is operated at a speed calculated using the decode timing information from the previous several frames [8]. The display buffer size for this scheme is 10.

• *Linear slack speed control* This is the enhanced feedback DVS control scheme proposed in this paper in which the decoding speed for the next frame is a linear function of the averaged available slack time as indicated by Equation (5). In our implementation, the averaging window size i is fixed to be 3, as it is a good trade-off between speed variation and system response time as discussed in Section 3. We also tested this technique with different display buffer sizes and found that a five-frame buffer would be a good design trade-off. Using $i = 3$, $B = 5$ and $u_{max} = 1.0, u_{min} = 0.435$ (the frequency range of our hardware platform), one can verify that both the stability condition (9) and real-time condition (10) are satisfied.

As revealed in Section 2, the time needed to display a

frame on the screen is insensitive to the CPU speed. Therefore, for the sake of fair energy consumption comparisons, in all of the above schemes, we scale the CPU speed to a fixed low frequency when the display process tries to send the decoded frame to the screen.

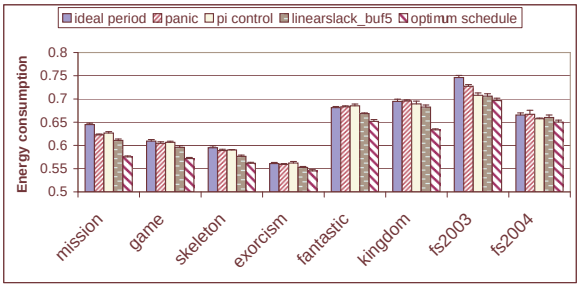


Figure 6. Total system energy consumption for video playback with different DVS schemes. The energy consumptions are normalized to those for the *full speed* schedule.

Figure 6 plots the measured total system energy consumption during video playback. As one might expect, all DVS schemes can save significant energy consumption compared to the case without DVS, and the *optimum* scheme achieves minimum energy consumption. The proposed *linear slack* scheme with buffer size 5 performs roughly equal to or better than the rest DVS schemes in terms of energy saving.

Table 1. Number of frames missing deadline with different DVS schemes

video clips	*panic* (buf5)	*linear slack* (buf5)	*PI controller* (buf10)
mission	94	69	38
game	53	34	19
skeleton	29	0	0
exorcism	10	0	0
fantastic	95	37	0
kingdom	77	4	0
fs2003	30	3	0
fs2004	172	6	0

In the theoretical analysis carried out in Section 3, we assume that the worst-case full speed frame decoding time is less than the display interval. However, during practical video play back, there are some frames exhibiting extremely long decoding times. Therefore, a significant number of frames might miss their deadlines. The Table 1 lists the number of frames missing their deadline in each video clip for different runtime DVS schemes. Though both schemes have the same display buffer size, the *linear slack* scheme does a much better job than the *panic* scheme to hide those extreme large frames. The *PI controller* scheme can further reduce the number of deadline misses at the cost of twice the buffer size. Given that the total number of frames in each clip is 3000, the deadline miss rate of the *linear slack* scheme is around 1% for most clips.

6. Conclusion

Although many low-power multimedia playback techniques have been proposed, many of them suffer from idealized assumptions or practical constraints. The work described in this paper seeks a balanced design by first creating a model for the available design space, then identifying a preferable architecture based on desirable system behaviors, and finally analyzing the design trade-offs using formal methods. The implementation overhead of this new DVS technique is ultra-low and can be easily implemented in both software and hardware based decoding systems.

We implemented the proposed architecture on a prototype hardware platform and compared it with existing techniques. Experimental results show that the proposed new design achieves equal or better energy efficiency than the existing techniques, brings close to ideal playback quality (about a 1% deadline miss rate), and only requires a moderate buffer size (5-frame buffer). Therefore, the proposed new DVS multimedia decoding architecture is indeed a good trade-off among energy, playback quality, hardware cost and playback latency.

References

[1] FFmpeg project. `http://ffmpeg.mplayerhq.hu/`.

[2] Movie trailers. `http://trailer.nerodigital.com/enu/index.html`.

[3] Movie trailers. `http://www.divx.com/movies`.

[4] Movie trailers. `http://www.apple.com/quicktime/guide/hd/`.

[5] 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.

[6] K. Choi, R. Soma, and M. Pedram. Off-chip latency-driven dynamic voltage and frequency scaling for an mpeg decoding. In *Proceedings of 41st Design Automation Conference*, pages 544 – 549, June 2004.

[7] C. Im and S. Ha. Dynamic voltage scheduling with buffers in low- power multimedia applications. *ACM Transactions on Embedded Computing Systems (TECS)*, 3(4):686–705, November 2004.

[8] Z. Lu, J. Lach, M. Stan, and K. Skadron. Reducing multimedia decode power using feedback control. In *Proc. of International Conference on Computer Design*, pages 489–96, October 2003.

[9] W. Rudin. *Real and Complex Analysis*. McGraw-Hill, 1987.

[10] Y. Tan, P. Malani, Q. Qiu, and Q. Wu. Workload prediction and dynamic voltage scaling for mpeg decoding. In *Proceedings of Asia and South Pacific Design Automation Conference*, pages 911–916, January 2006.