A Comparative Study of Dynamic Voltage Scaling Techniques for Low-Power Video Decoding

Eriko Nurvitadhi and Ben Lee
School of Electrical Engineering and Computer Science
Oregon State University
{nurviter, benl}@eecs.orst.edu

Chansu Yu
Department of Electrical and Computer Engineering
Cleveland State University
c.yu91@csuohio.edu

Myungchul Kim
School of Engineering Information and Communications University
Taejon, Korea
mckim@icu.ac.kr

Abstract

This paper presents a comparison of power-aware video decoding techniques that utilize Dynamic Voltage Scaling (DVS) capability. Three techniques were simulated and compared in terms of power consumption, accuracy, and deadline misses. The simulation results show that the dynamic per-frame technique, where the decoding time prediction adapts to the particular video being decoded, is the most promising approach due to its feasibility of implementation and comparable performance to the ideal case. Our findings also indicate that in general, as the number of available processor settings increases, the amount of power saving increases, but the number of deadline misses increases as well. More importantly, most of these deadline misses are within 10-20% of the playout interval and thus insignificant. However, video clips with high variability in frame complexities combined with inaccurate decoding time predictions may degrade the video quality. Finally, our results show that a processor with 13 voltage/frequency settings is sufficient to achieve near maximum performance with the experimental environment and the video workloads we have used.

Keywords: *Dynamic voltage scaling, video decoding, low-power techniques, decoding time prediction.*

1. Introduction

Power efficient design is one of the most important goals for mobile devices, such as PDAs, handhelds, and mobile phones. As the popularity of multimedia applications for these portable devices increases, reducing their power consumption will become increasingly important. Among multimedia applications, delivering video will become the most challenging and important applications of future mobile devices. Video conferencing and multimedia broadcasting are already becoming more common, especially in conjunction with the Third Generation (3G) wireless network initiative [8]. However, video decoding is a computationally intensive, power ravenous process. In addition, due to different frame types and variation between scenes, there is a great degree of variance in processing requirements during execution. This high variability in video streams can be exploited to reduce power consumption of the processor during video decoding.

Dynamic Voltage Scaling (DVS) has been shown to take advantage of the high variability in processing requirements by varying the processor's operating voltage and frequency during run time [4, 7]. In particular, DVS is suitable for eliminating idle times during low workload periods. Recently, researchers have attempted to apply DVS to video decoding to reduce power [9, 10, 12, 13]. These studies present approaches that predict the decoding times of incoming frames or Group of Pictures (GOPs), and reduce or increase the processor setting based on this prediction. As a result, idle processing time, which occurs when a specific frame decoding completes earlier than its playout time, is minimized.

Even if decoding time prediction is very accurate, the maximum DVS performance can be achieved only if the processor can scale to very precise processor settings. Unfortunately, such a processor design is impractical since there is cost associated with having different processor supply voltages. Moreover, the granularity of voltage/frequency settings induces a tradeoff between power savings and deadline misses. For example, the fine-grain processor settings may even increase the number of deadline misses when it is used with an inaccurate decoding time predictor. Coarse-grain processor settings, on the other hand, lead to overestimation by having voltage and frequency set a bit higher than required. This reduces deadline misses in spite of prediction errors, but at the cost of reduced power savings. Therefore, the impact of available processor settings on video decoding with DVS needs to be further investigated.

Based on the aforementioned discussion, this paper provides a comparative study of the existing DVS techniques developed for low-power video decoding, such as GOP [12] and Direct [10, 13], with respect to prediction accuracy and the corresponding impact on performance.

In addition, an alternative method called Dynamic is proposed as an improvement to these techniques. The Dynamic approach is designed to perform well even with high-motion videos by dynamically adapting its prediction model based on the decoding experience of the particular video clip being played. An extensive simulation study based on SimpleScalar processor model [5], Wattch power tool [3] and Berkeley MPEG Player [2] has been conducted to compare these DVS approaches. We investigated two important tradeoffs: The impact of decoding time predictions and granularity of processor settings on DVS performance in terms of power savings, playout accuracy, and characteristics of deadline misses.

The rest of the paper is organized as follows. Section 2 presents existing DVS techniques on low-power video decoding and their decoding time predictors. Section 3 discusses the simulation environment and presents the simulation results on how the accuracy of decoding time predictor and the granularity of processor settings affect DVS performance. Finally, Section 4 provides a conclusion and elaborates on future work.

2. Prediction-based DVS Approaches

Prediction algorithms employed in several DVS approaches differ based on the following two criteria: Prediction interval and prediction mechanism. *Prediction interval* refers to how often predictions are made and processor settings are changed. The existing approaches use either per-frame or per-GOP scaling. *Prediction mechanism* refers to the way the decoding time of an incoming frame or GOP is estimated. Currently, all the approaches utilize some form of frame size vs. decoding time relationship [1]. Some methods are based on a fixed relationship, while others use a dynamically changing relationship. In the fixed approach, a linear equation describing the relationship between frame sizes and frame decoding times is provided ahead of time. In the dynamic approach, the frame-size/decoding-time relationship is dynamically adjusted based on the actual frame sizes and decoding times of a video stream being played. The dynamic approach is better for high-motion videos where the workload variability is extremely high. In other cases, the fixed approach performs better than the dynamic approach but its practical value is limited because the relationship is not usually available before actually decoding the stream.

Three DVS techniques for video decoding and the corresponding prediction algorithms are discussed and compared: GOP is a per-GOP and dynamic approach, Direct is a per-frame and fixed approach, and Dynamic is a per-frame and dynamic approach. Another proposed method completely bypasses the decoding time prediction to eliminate the possibilities of errors due to inaccurate scaling predictions [9]. This is done by preprocessing video streams to add accurate video complexity information during the encoding process. However, this approach is impractical since it requires modification in the widely used standard video stream format. We do not include this method in the study.

2.1. Per-GOP Approach with Dynamic Equation (GOP)

GOP dynamically recalculates the slope of the frame-size/decode-time relationship based on the decoding times and sizes of past frames [12]. At the beginning of a GOP, the sizes and types of the frames of an incoming GOP are observed. This information is then applied to the frame-size/decode-time model, and the time needed to decode the GOP is estimated. Based on this estimate, the lowest frequency and voltage setting that would satisfy the frame rate requirement is selected. The dynamic slope adjustment was originally presented by Bavier *et al*. [1]. Here, the slope adjustment is implemented by utilizing the concept of Decoding Time Per Byte (DTPB). DTPB essentially represents the slope of the frame-size/decode-time equation and this value is updated as the video is decoded using the actual decoding times of the just-decoded frames.

2.2. Per-Frame Approach with Fixed Equation (Direct)

Direct was used by Pouwelse *et al.* in their implementation of StrongARM based system for power-aware video decoding [10, 13]. In this technique, the scaling decision is made on a per-frame basis. Based on a given linear model between frame sizes and decoding times, decoding time of a new frame is estimated and then it is associated to a particular processor setting using a direct mapping.

In order to obtain the frame size of the new frame, the video decoder examines the first half of the frame as it is being decoded. Then, the size of the second half of the frame is predicted by multiplying the size of the first half with the complexity ratio between the first and second halves of the previous frame. If the decoding time of the first half of the frame is higher than half of the estimated decoding time, it means that the decoding is too slow; the processor setting is then increased.

In addition, they present a case in which the frame sizes are known a priori [13]. This is achieved by feeding the algorithm with the size of each frames gathered offline. Thus, voltage/frequency scaling is done at the beginning of each frame by looking at the frame size, estimating the decoding time, and scaling the processor setting accordingly. Our simulation study of Direct is based on this case.

2.3. Per-Frame Approach with Dynamic Equation (Dynamic)

Dynamic is a per-frame scaling method that dynamically updates the frame-size/decoding-time model and the weighted average decoding time.

The mechanism used to dynamically adjust the frame-size/decode-time relationship is similar to one presented by Bavier *et al*. [1]. In Dynamic, the adjustment is made by focusing on the differences of the decoding times and frame sizes. The average decoding time of previous frames of the same type is used as the initial value for predicting the next frame. The possible deviation from this average value is then predicted by looking at the weighted difference of frame sizes and decoding times of previous frames. This predicted fluctuation time is then added to the average decoding time to obtain the predicted decoding time of the incoming frame.

3. Performance Evaluation and Discussions

In this section, we present the simulation results comparing the performances of different DVS schemes introduced in Section 2, and also show the quantitative results on the effect of different processor setting granularities. Performance measures are average power consumption per frame (APF), error rate, and deadline misses. Before proceeding, the simulation environment and the workload video streams are first described in Subsections 3.1 and 3.2, respectively.

3.1. Simulation Environment

Our simulation environment consists of modified SimpleScalar [5], Wattch [3], and Berkeley MPEG Player [2]. SimpleScalar is used as the basis of our simulation framework. The simulator is configured to resemble the characteristics of a five-stage pipeline architecture, which is typical of processors used in current portable multimedia devices. The proxy system call handler was modified and a system call for handling voltage and frequency scaling was added. Thus, the MPEG decoder makes a DVS system call to the simulator to adjust the processor setting.

For video decoding, the Berkeley MPEG decoder was modified to make DVS system calls. A DVS system call modifies the voltage and frequency values presently used by SimpleScalar. Therefore, before processing a frame, a system call to the simulator is performed to start the power consumption count and to obtain the time at which the decoding started. After the decoding process completes, another system call is made to stop the power consumption count. Finally, at the time when the frame is supposed to be displayed, another system call is issued to indicate the frame display time. This information is needed to check whether the frame missed its playout time due to miss-prediction, and by how much.

For the simulation study, the overhead of processor scaling was assumed to be negligible. In practice, there is a little overhead related with the scaling. Previously implemented DVS systems have showed that processor scaling takes about 70 μs to 140 μs [4, 10, 13]. Since this overhead is significantly smaller than the granularity in which the DVS system calls are made, they have a negligible effect on the overall results.

Table 1: The characteristics of the clips used in the simulation.

Clip Name	*Children*	*Red's Nightmare*	*Under Siege*
Type	Low-motion	Animation	High-motion
fps	29.97	25	30
I	62	41	123
P	238	81	122
B	599	1089	486
Total	899	1211	731
Size	320 x 240	320 x 240	352 x 240
Time-Size Equation	Decoding time = 88.8 × Frame size + 10^6	Decoding time = 53.9 × Frame size + 2×10^6	Decoding time = 69.6 × Frame size + 2×10^6
R^2	0.94	0.89	0.94

3.2. Workload Video Streams

Three MPEG clips were used in our simulations. These clips were chosen as representatives of three types of videos - low-motion, animation, and high-motion. A clip showing a public message on child care is selected for a low-motion video (*Children*) and a clip named *Red's Nightmare* is selected as an animation video. Lastly, a clip from the action movie *Under Siege* is selected to represent a high-motion video. Table 1 shows the characteristics of the clips, including frame-size/decode-time equations, which were generated after preprocessing each clip. The R^2 coefficient represents the accuracy of the linear equations, i.e., the closer R^2 is to unity, the more likely the data points will lay on the predicted line.

3.3. Prediction Accuracy on DVS Performance

Figures 1 and 2 summarize power savings and error results. The ideal case (Ideal) was also included as a reference. The ideal case represents perfect prediction with voltage/frequency set to any accuracy required, and thus represents optimum DVS performance. This was done by using previously gathered actual frame decoding times to make scaling decisions.

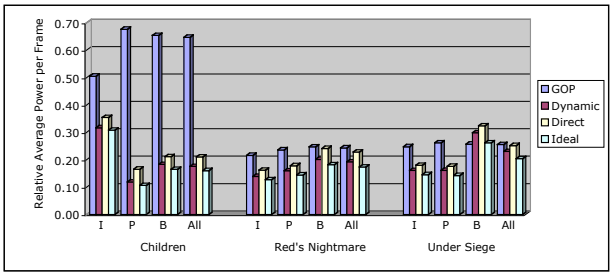

Figure 1: Relative APF for various DVS approaches.

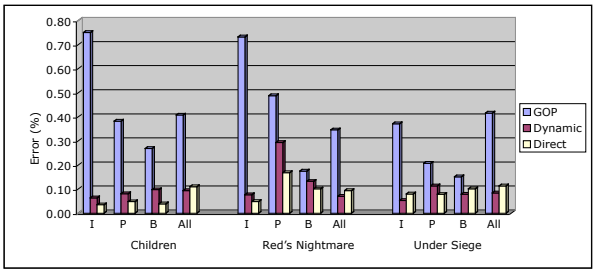

Figure 2: Errors for various DVS approaches.

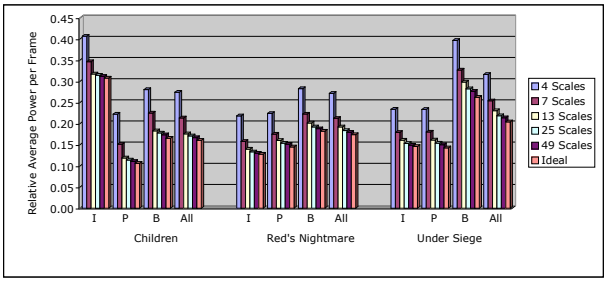

Figure 3: Relative APF for various settings.

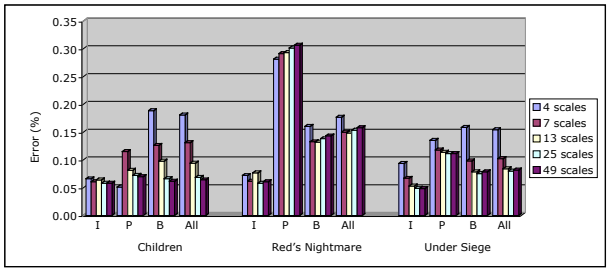

Figure 4: Errors for various settings.

Figure 1 shows the power savings in terms of average power consumption per frame relative to using no DVS for all frames as well as for each frame type. All three approaches achieve comparable power savings to the ideal case, except GOP with *Children* (i.e., only 35% improvement). GOP performs the worst because it applies the same processor setting over multiple types of frames in a GOP. This consequently wastes the potential power savings that can be made for P- and B-frames, which typically have shorter decoding times than I-frames. On the average, Dynamic provides the most power saving (80% improvement) but it is only slightly better than Direct (77%).

Figure 2 shows the accuracy of the three DVS approaches in terms of error, defined as the ratio of standard deviation of inter-frame playout times [14] to playout interval. GOP has the highest overall average error (39.3%) for the three clips. Dynamic was the most accurate with average error of 8%, closely followed by Direct with 10.5%. Neglecting the error results of GOP, the amount of error for each frame type depends heavily on the variability of decoding times for Direct and Dynamic. For example, for *Red's Nightmare*, both P-and B-frames resulted in significant errors (17% and 10% for Direct, and 29% and 13% for Dynamic). This was also the case for P-frames in *Under Siege*.

3.4. Impact of Processor Settings Granularity

The results of power consumption and accuracy presented in the previous subsection were based on 13 frequency/voltage settings. Thus, even if very accurate decoding time predictions are made, the granularity of voltage/frequency settings will invariably affect the performance of DVS. It seems that having fine-grain voltage scales would lead to better performance than having coarse-grain scales. Nevertheless, a clearer understanding is needed about the impact of various processor scaling granularities have on video decoding in terms of power consumption and accuracy.

To show the aforementioned tradeoff, we experimented with various scaling schemes consisting of 4, 7, 13, 25, and 49 scales with the minimum setting of 0.79 V, 59 MHz and maximum of 1.65 V, 251 MHz. Each of these schemes was simulated using the Dynamic approach. This approach was chosen as a representative among others due to its promising performance and high potential for realistic implementation.

The results are shown in Figures 3 and 4. Figure 3 shows the relative average power consumption per frame compared to using no DVS. As can be seen, power consumption decreases as the number of processor settings increase. However, power saving only increases slightly beyond 13 scales. Thus, using 13 available settings seems to be sufficient to achieve relative average power per frame comparable to the ideal case (e.g., 18% vs. 16% for Children, 19% vs. 17% for *Red's Nightmare*, and 23% vs. 20% for *Under Siege*, respectively).

Figure 4 shows the accuracy of the DVS approaches for the various settings. In general, the error decreases with the availability of more processor settings. This is true for *Children* and *Under Siege*, where changing the available number of processor settings from 4 to 49 settings significantly reduces the error. However, this is not the case for *Red's Nightmare*. The error decreases for the processor settings of 4 to 13, but for the number

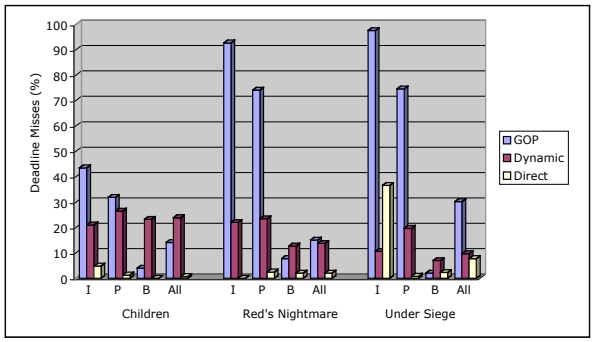

Figure 5: % of deadline misses for various DVS approaches.

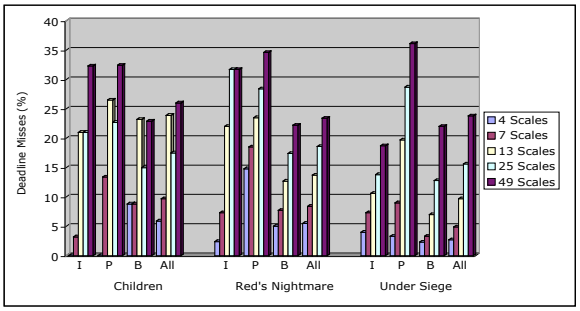

Figure 6: % of deadlines misses for different settings.

of settings more than 13, the ratio increases slightly due to large errors for P- and B-frames. Therefore, with the finer granularity, more of the inaccuracies are getting scaled more precisely (e.g., propagated).

3.5. Characteristics of Deadline Misses

Figure 5 shows the deadline misses for the three DVS approaches. As can be seen, the Direct approach resulted in the smallest percentage of deadline misses. This is because we are using a frame-size/decoding-time equation that is based on the specific characteristic of each clip. Thus, the frame-size/decoding-time model is well suited for the particular clip being run. For Direct, the *Under Siege* clip resulted in the most number of misses (7.8%). The reason is that the clip is a high-motion video, which deviates most from the calculated linear model.

However, the Dynamic approach handles the *Under Siege* clip comparatively well (9.7% deadline misses) because its adaptive capability in predicting decoding times. The *Children* clip resulted in the most number of misses (23.92%) for Dynamic. Unlike the other methods, the highest number of deadline misses occurred for the clip with the least amount of scene variations. This is because the dynamic decoding time estimation used performs too aggressively for the clip that has smooth movement. GOP also uses an adaptive mechanism similar to the Dynamic approach. However, the deadline misses are minimized by having longer scaling intervals (i.e., per-GOP instead of per-frame). Moreover, its scaling decision includes all types of frames. Thus, P- and B-frames, which typically have shorter decoding times than I-frames, would likely be overestimated since the setting used has to also satisfy the playout times for I-frames.

Figure 6 shows the deadline misses for various granularity voltage/frequency scales using Dynamic. The number of deadline misses increases linearly as the granularity of the processor settings becomes finer, except for the *Children* clip. This is because by having more available settings, the scaling decisions rely more on the estimation of the decoding times. Thus, an estimation error would easily propagate to cause a deadline miss. Essentially, the main factor that would affect the relationship between the granularity of the processor scale and DVS performance is the distribution of the frame decoding times. The power savings and deadline misses would depend on whether the processor settings available could satisfy the expansion of the decoding times to the frame playout intervals.

Figure 7 shows the characteristics of deadline misses in terms of how much the desired playout times were exceeded for various DVS approaches and different processor settings. The x-axis shows the extent of the deadlines misses relative to the playout interval, categorized as 10%, 20%, 30%, 40%, and greater than 40%.

The y-axis represents the percentage of deadline misses over an entire clip. For example, a 5% value on the y-axis with the 10% category on the x-axis means that 5% of the frames in the clip that miss the deadline missed it by 10% of the desired playout interval (e.g., for 25 fps, or 40 ms playout interval, these frames are played out between 40 to 44 ms after the preceding frames). The z-axis categorizes the results for different clips and frame types.

For the Direct and Dynamic approaches, most of the misses are within 10% of the playout interval. In addition, virtually all of the misses for three approaches lie within the 20% range. For GOP, the deadline misses are more erratic since they are concentrated in 10% and >40% ranges. Thus, deadline misses in the GOP scheme have a higher potential of disrupting the quality of video playback. Conversely, deadline misses in Direct and Dynamic are less likely to affect the quality of the videos [6]. Additionally, even though the number of deadline misses increases as more processor settings are made available, the extent to which these deadlines are missed remains relatively constant within the 20% range for all the clips. Moreover, most of the misses are still within the 10% range.

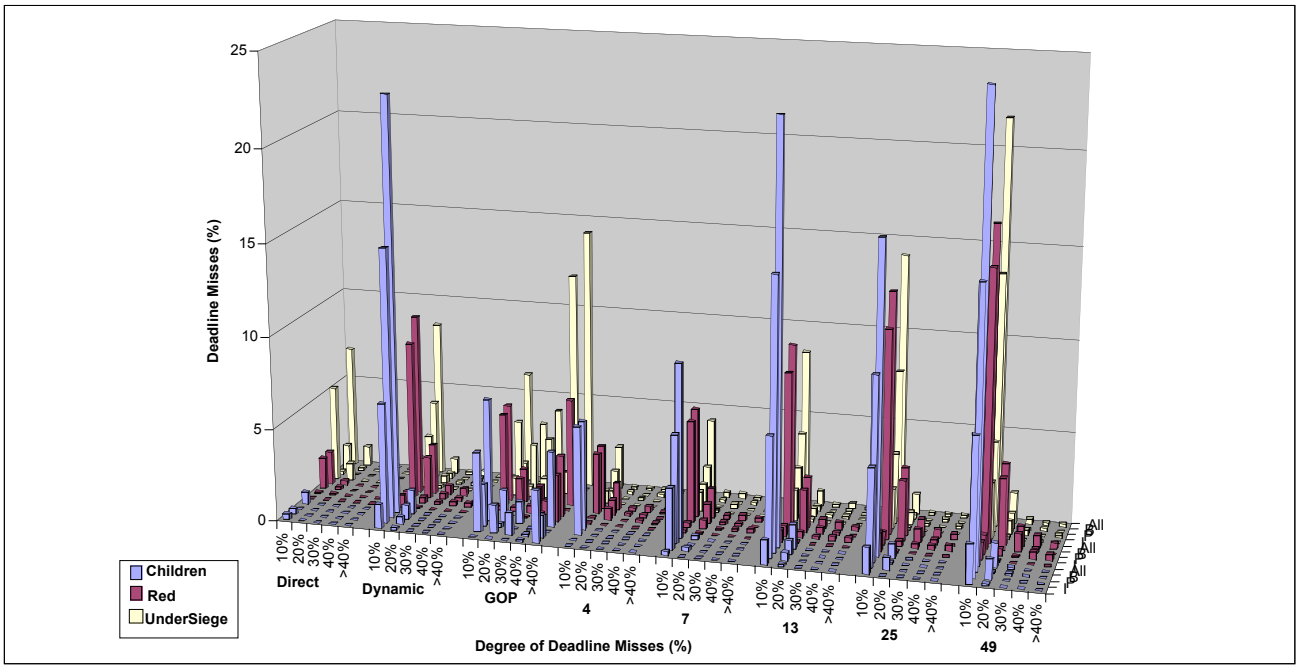

Figure 7: The degree of deadline misses for various DVS approaches and different processor settings.

4. Conclusion

This study compared the existing DVS techniques for low-power video decoding. Out of the three approaches simulated, Dynamic and Direct provided the most power savings. Among the two, Dynamic is much more practical, because it is able to dynamically adapt to any video stream. The implementation of such an approach would not change any external behavior of the system. Thus, this approach is very suitable for portable multimedia devices, which require low-power consumption.

Our study also further quantifies the deadline misses that occurred by looking at the degree to which the play-out times are exceeded. The results indicate that, for the Dynamic and Direct approaches, most of the deadline misses are within 20% of the playback interval. Therefore, it is less likely to degrade the quality of the video. In addition, in designing a DVS capable processor for video decoding, higher number of processor settings is preferable. By having more available settings, more power saving can be achieved without any additional risk of sacrificing quality of the video. More deadline misses may occur, but they are still within a tolerable range [6].

As future work, it would be interesting to investigate the usage of DVS system on streaming video. In such a case, packet jitters from the network also need to be considered. It would also be useful to assess the cost of implementing very fine-grain scales on DVS processor. Finding more accurate prediction mechanisms and new ways to exploit DVS for low power video decoding is also critical. However, our simulation results show that the existing approaches have already reached near-maximum performance (and similar conclusion has been made in [10]), but opportunity for improvement still exists. Lastly, it would be beneficial to find ways to use DVS on other parts of a system, such as applying DVS to memory or network interface.

5. References

[1] A. Bavier, B. Montz, and L. Peterson, "Predicting MPEG Execution Times," *SIGMETRICS /PERFORMANCE '98, International Conference on Measurement and Modeling of Computer Systems*, pp. 131-140, June 1998.

[2] Berkeley MPEG Tools. http://bmrc.berkeley.edu/frame/research/mpeg/

[3] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A Framework for Architectural-Level Power Analysis and Optimizations," *Proceedings of the 27th International Symposium on Computer Architecture*, pp. 83-94, June 2000.

[4] T. D. Burd, T. A. Perimg, A. J. Stratakos, and R.W. Brodersen, "A Dynamic Voltage Scaled Microprocessor System," *IEEE Journal of Solid-State Circuits*, November 2000.

[5] D. Burger and T. M. Austin, "The SimpleScalar Tool Set, Version 2.0," *CSD Technical Report #1342*, University of Wisconsin, Madison, June 1997.

[6] M. Claypool and J. Tanner, "The Effects of Jitter on the Perceptual Quality of Video," *In Proceedings of the ACM Multimedia Conference, Vol. 2*, Orlando, USA, Florida,

November 1999.

[7] K. Govil, E. Chan, and H. Wasserman, "Comparing Algorithms for Dynamic Speed-Setting of a Low Power CPU," *Proc. 1st Int'l Conference on Mobile Computing and Networking*, Nov. 1995.

[8] L. Harte, R. Levine, and R. Kikta, "3G Wireless Demystified," McGraw-Hill, 2002.

[9] M. Mesarina and Y. Turner, "Reduced Energy Decoding of MPEG Streams," *ACM/SPIE Multimedia Computing and Networking 2002* (MMCN '02), San Jose, CA, 18-25 January 2002.

[10] J. Pouwelse, K. Langendoen, R. Lagendijk, and H. Sips, "Power-Aware Video Decoding," *Picture Coding Symposium* (PCS '01), Seoul, Korea, April 2001.

[11] J. Shin, "Real Time Content Based Dynamic Voltage Scaling", *Master Thesis*, Information and Communication University, Korea, January, 2002.

[12] D. Son, C. Yu, and H. Kim, "Dynamic Voltage Scaling on MPEG Decoding," *International Conference of Parallel and Distributed System (ICPADS)*, June 2001

[13] J. Pouwelse, K. Langendoen, and H. Sips, "Dynamic Voltage Scaling on a Low-Power Microprocessor," *7th ACM Int. Conf. on Mobile Computing and Networking (Mobicom)*, pp. 251-259, Rome, Italy, July 2001.

[14] Y. Wang, M. Claypool, and Z. Zuo, "An Empirical Study of RealVideo Performance Across the Internet," *Proceedings of the First Internet Measurement Workshop*, pp. 295-309, November 2001.