Research Journal of Information Technology 1(1): 17-21, 2009
ISSN: 2041-3114

Submit Date: April 28, 2009 Accepted Date: May 27, 2009 Published Date: August 29, 2009

Low Power MPEG Video Player Using Dynamic Voltage Scaling

[1]A. Rajalingam, [1]B. Kokila and [2]S.K. Srivatsa
[1]Vel Tech Multi Tech Dr.Rangarajan Dr.Sakunthala Engineering College, Chennai, India.
[2]Anna University, Chitlapakkam, Chennai ,India.

Abstract: In static voltage mechanism MPEG video player, takes more energy. In our paper dynamic voltage/frequency scaling (DVS) mechanism is implemented and our player scales the voltage level for the frames of next interval based on decoding time prediction. The decoding time of a frame is predicted by the size and the type of the frame. We implemented our mechanism by modifying a traditional MPEG1 decoder. Our experiments show that a significant amount of processor energy savings can be achieved while enhancing QoS of video playback.

Key words: DVS (Dynamic Voltage Scaling), Low Power VLSI, MPEG.

INTRODUCTION

Most of the mobile devices have energy constraints, embodied by a battery that has a finite lifetime. Among the devices in mobile systems, processor becomes critical in energy efficiency as computation requirement for a mobile system increases. The most effective way to reduce power consumption of a processor core in CMOS technology is to lower the supply voltage, which exploits the quadratic dependence of power on voltage. Reducing the supply voltage, however, increases circuit delay and decreases clock speed. This may not be effective because some systems have latency critical tasks. One possible solution to this problem is to dynamically vary the frequency/voltage (Dynamic Voltage Scaling, DVS) according to the processor workload, which is studied extensively. Recent advances in power supply technology make it possible to create processor cores with varying supply voltages according to application's time constraints (1). Current custom and commercial CMOS chips are capable of operating reliably over a range of supply voltages, and efficient variable voltage DC suppliers are available.

DVS allows a processor to dynamically changes its speed and voltage at run time, increasing energy efficiency. In (2-4), interval based DVS has been proposed, which divides time into uniform-length intervals and analyzes system utilization of the previous intervals to determine the voltage of the next interval accordingly. Though interval based scheduling is simple and easy to implement, it often incorrectly predicts future workloads and degrades the quality of service. To DVS be more efficient, application may direct DVS (Application directed DVS), as an application can be the best place that can provide information of future workload (14). On the other hand, MPEG player is one of most important killer applications in mobile environment. Though it requires considerable computing power and energy, adapting DVS on it has two obstacles. One is that the workload of MPEG decoding varies widely due in part to the fact that a given MPEG video stream contains different frame types, and in part to the potential wide variation between scenes, and the other is that it should conform real-time constraints. These two hurdles make application directed DVS with predicted MPEG decoding time (DT) crucial for both energy efficiency and real-time conformity of MPEG playback. Our MPEG player provides predicted MPEG DT of future interval, with which it scales frequency/voltage to minimize power consumption while enhancing the QoS of real-time playback.

Background and Related Works

Problem Description: In a digital CMOS circuit, power consumed by each operation is given by the following equation (3)

$$P \alpha C_{eff} V^2 f_{CLK}$$

where Ceff is the effective switching capacitance of the operation, V is the supply voltage, and f_{CLK} is the clock frequency. On the other hand, delay of the circuit and the clock frequency are given by:

$$delay \alpha \frac{V}{(V-Vk)^2} \quad and \quad fCLK \alpha \frac{(V-Vk)^2}{V}$$

where α ranges from 1 to 2, and Vk depends on threshold voltage at which velocity saturation occurs (3). Regardless the exact value of Vk, the combination of the two equations implies that lowering the supply voltage will result in longer delay, but at the same time quadratic decrease in energy. In reality, it is more than quadratic since the clock frequency can also be reduced when the supply voltage is scaled down. Current VLSI technology supports a set of supply voltage levels. For example, a *Motorola CMOS 6805* micro-controller is rated at 6Mhz at 5.0 volts,4.5Mhz at 3.3 volts, and 3Mhz at 2.2 volts (2).

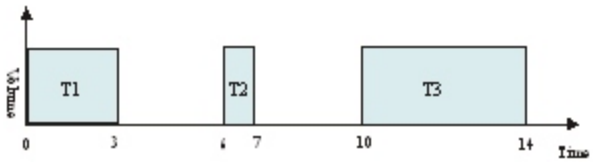
Corresponding Author: A.Rajalingam, Vel Tech Multi Tech Dr.Rangarajan Dr.Sakunthala Engineering College, Chennai, India

More recently, Crusoe processor from Transmeta has 16 voltage levels and LongRun power management is provided to scale voltage on application side (5). This contrasts to other power saving schemes where the operating frequency is changed instead of the supply voltage. If only the frequency were changed, instantaneous energy consumption is altered, but the total energy consumed for each operation remains the same.

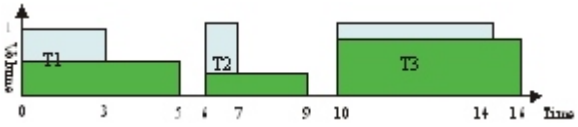
Dynamic Voltage Scaling (DVS): Dynamic Voltage Scaling (DVS) is a technique that allows a voltage scheduler to alter a microprocessor's operating voltage at run-time and thus trade-off energy for delay (2,3). Since the workload is usually not constant but in bursts, there are idle times in which CPU does not work. One conventional way of reducing CPU power consumption is shutdown mechanism, which stops CPU operation while idling. Comparing with DVS, it is less effective since the energy is a quadratic function of voltage. Fig. 1(a) shows an example of the voltage-scheduling graph with the shutdown mechanism. The X-axis represents time and the Y-axis represents the voltage and processor speed. We assume that the voltage and frequency are exactly proportional. In the figure, the computation required by a task is represented by the 'area under the curve' for that task. Voltage and processor speed is normalized to the range (0…1). The height of a task (voltage or processor speed) and the running time determine the energy required to complete the task. Assuming that three jobs (T1, T2 and T3) in Fig. 1 (a) are executed with full voltage (1 volt for example) and the processor becomes shutdown when idle, the energy consumed for the three jobs is 8 energy units. With DVS, the supply voltage can be reduced to 3/5, 1/3 and 2/3 for the three jobs, respectively, which results in the same amount of computation (the same area under the curve) but they require less energy consumption of only 2.97 energy units as shown in Fig. 1(b).

Since the dynamic voltage scaling relies on system workload, information of future workload is critical to handle DVS efficiently. But it is not usually possible to know exact future workload of non-real time systems.

MPEG Decoding: This section discusses the basic structure of MPEG stream as well as DT and decoding strategy for real-time playback. The MPEG video compression standard [6] defines a video stream as a sequence of still images or frames. A standard MPEG stream is composed of three types of compressed frames: I, P and B. While I frames are intra-coded, the generation of P and B frames involves, in addition to intra-coding, the use of motion prediction and interpolation techniques. As a result, I frames are, on the average, the largest in size, followed by P frames, and finally B frames. A Group of Pictures (GOP) is a sequence of frames from one I frame to the next I frame. Encoders use a fixed GOP pattern when compressing a video sequence, where the GOP pattern specifies the number and temporal order of



(a) CPU workload with shutdown mechanism



(b) CPU workload with Dynamic Voltage Scaling

Fig 1: Voltage scheduling graphs by two mechanisms

P and B frames between two successive I frames. Each video has different GOP sequences and numbers (7).

Another characteristic of a MPEG decoder is its real-time constraints. Frames should be passed with a given frames per second (fps) either played or dropped. MPEG players should check at a certain point during playback if it should drop frames or sleep to maintain real-time playback. The checkpoint depends on implementation. There are two common ways to check. One is frame based real-time playback which checks every end of a frame decoding and determine if next frame be dropped or delay decoding till next start of frame decoding. The other scheme is interrupt based real-time playback which usually interrupts during MPEG play-back periodically to determine if required frames are passed after previous interrupt occurred. In case of interrupt based real-time playback, the player should monitor the current system speed with real-time clock, and decide each frame either to drop or to play. It usually maintains four real-time phases which are categorized by relative system speed to frame decoding, namely, DELAYPHASE, B-PHASE, P-PHASE and I-PHASE (8). If the system is too fast to meet the real-time constraint, the decoder sets mode as DELAY-PHASE which delays between frames according to the delay value. The delay value shows how long the player delay till next frame decoding to meet real-time constraints, that is, how fast the system speed is. On the contrary, if the system is unable to play all the frames, then a drop rate is set and some of the B frames are discarded (B-PHASE) by the drop rate. If the system is still too slow even all the B frames are skipped, then some of the P frames are dropped (P-PHASE). Finally, I frames can be dropped in the worst case (I-PHASE). Since I frames and P frames are referred by P or B frames, dropping I or P frames causes other frames referring those to be dropped also. Determining delay value and frames drop can be done every n time. The n is usually 1 second. From DT point of view, I frames usually take longer than other frames while B frames take the least. On the contrary, I-frame decoding time per byte (I_DTPB) is the shortest while B-frame decoding time per byte (B_DTPB)

is the longest. This is because a large computational work is needed in motion prediction in case of B and P frames. Therefore, the frame DT can be predicted by frame type and size, and the corresponding predictor is shown to have less than 25% of prediction error (9). In the next section, we exploit the predictor to develop our proposed DVS algorithm.

DVS for MPEG decoding: There has been several works on reducing power consumption in MPEG decoding with DVS. In (10), a method using feedback control was proposed, in which macro blocks in a frame are first divided into two parts, and the DT of the second part is predicted based on DT and the code size of the first part. If the DT of the first part exceeds the predicted DT, the voltage for the next part is increased so that a deadline violation can be avoided. This technique thus scales up the voltage and frequency during the second part to meet the deadline when a prediction error occurs in the first part, thus results in higher energy consumption for performing the IDCT step in the second part when the prediction error occurs in the first part. [11] solved this problem by dividing decoding process into frame dependant and frame independent, thus using frame independent part as a timing buffer when misprediction occurs. But these two researches scales frequency and voltage at every frame decoding, which can cause overhead both on time and energy if fps is a big number. In [12], a table, which contains information of DT of frames, is maintained and provided by a contents provider for end-user to play MPEG with DVS. In which prediction of MPEG DT is performed at every start of GOP. The problem of this algorithm is that it causes QoS degradation as the scaling activity is not performed at the time that real-time conformity check occurs.

DVS Algorithms for Real-time MPEG Playback: In this section, we present two algorithms that minimize energy consumption while enhancing realtime conformity on MPEG playback. The first DVS algorithm is based on previous workload, while the second one uses predicted MPEG DT as well as previous workload.

DVS based on Delay Value for Real-time Playback (DDVS): To conform real-time constraint, the MPEG player maintains delay value for delay phase and drop rate for I, P and B phase. When the phase is I, P or B phase, maximum processor power is needed, thus no voltage scaling will occur. Therefore, we scale voltage and frequency only on delay phase. The delay value implies that playback should pause as much as delay value till next interrupt to meet real-time playback. The player sleeps same amount of time which is the delay value divided by fps after each frame decoding. Our first algorithm (DDVS, Delay DVS) scales frequency and voltage according to delay value to conform real-time playback until next interrupt instead of sleeping at every frame decoding. For example, if interrupt to check real-time playback occurs at every 1 second, and the delay value is set to be 0.3 second, we can set frequency and voltage as 70% of maximum value instead of sleeping 0.3 second during next real-time playback.

Algorithm for DVS based on delay value (DDVS)

```
...../*we are in real-time check phase*/
delay =set_delay(prev_played_frames – fps);
if(delay > 0){ //delay phase
next_freq = 1-delay;
}
else next_freq = 1;
set_volt_freq(next_freq);
.....
```

This simple DVS algorithm is a kind of interval based DVS in a sense that delay value is a result of previous history. Therefore, if DT of next period varies much compared with previous interval, real-time conformity and QoS of video may decrease. Our next algorithm solved this problem by considering predicted MPEG frame DT.

DVS based on Predicted MPEG DT for Real-time Playback (PDVS): High fluctuation, which degrades QoS of playback with DDVS algorithm, is due in part to the fact that a given MPEG video stream contains different frame types, and in part to the potential wide variation between scenes (e.g., talking heads versus action). This makes the prediction of DT based on the past behavior difficult. We propose an efficient DVS algorithm for real-time playback which uses both the previous history and the estimated DT which is called prediction based DVS (PDVS, Prediction DVS). DT is estimated at every interrupts by considering MPEG frame types and sizes as well as previous workload information. Moreover, these values differ by scene characteristics and vary throughout the MPEG decoding.

Algorithm for DVS based on predicted MPEG DT

```
...../*we are in real-time check phase*/
Updata_statistics();
next_dec_time= get_next_I_size() * avg_I_DTPB
+ get_next_P_size() * avg_P_DTPB
+ get_next_B_size() * avg_B_DTPB;
if(next_dec_time < time_interval){
next_freq = next_dec_time / time_interval;
}
else next_freq = 1;
set_volt_freq(next_freq);
.....
void update_statistics( )
{
/* weighted average */
avg_I_DTPB=avg_I_DTPB   *(1-weight_factor)   +
I_DTPB * weight_factor;
avg_P_DTPB=avg_P_DTPB   *(1-weight_factor)   +
P_DTPB * weight_factor;
avg_B_DTPB=avg_B_DTPB   *(1-weight_factor)   +
B_DTPB * weight_factor;
}
```

To adapt to workload changes and scene characteristics, we maintains DTPBs by weighted sum of previous average as shown in Algorithm. The weight_factor controls the relative weight of the most recent and past history of DTPBs. This number is set to a specific number (0.3 in our simulation with which the results are the best). But, as the best weight_factor is stream specific, no optimal weight_factor for all streams exists. We then get *I*, *P* and *B*-frame sizes of next interval and estimate the next DT by multiplying frame sizes and DTBP values. Finally the voltage and frequency for the next interval are set according to the estimated DT. For example, if estimated DT is 0.6 second and the interval is 1 second, the frequency and voltage will be set to be 0.6.

Performance Evaluation

Experimental Environment: In this section, we briefly overview the MPEG decoders as well as the video streams used in our experimental study. We also present the assumptions we made for simplifying the model.

MPEG Streams: We selected six famous movies for evaluating our decoders with respect to energy consumption. Instead of playing the whole movie, we used a portion (3000 frames) from each movie. We characterize the six movies with respect to average frame size, average DT, deviation of GOP DT and fluctuation of frame DT as shown in Table 1. Fluctuation is a degree of differences between adjacent frames DTs. High fluctuation leads to high errors in predicting the GOP DT. Action movies, such as *Alien* and *Angel* in the sample streams, usually have large fluctuations.

As we have no hardware platform that supports DVS, we assumed as listed below for our experimental study. Firstly, we just calculated energy consumption from experiments without actual voltage scaling. By this assumption, the value of frequency and voltage can be anything in continuous area from 0.5 to 1. This is reasonable as current processors are more and more increasing the number of possible voltage levels, which can be seen that Crusoe from Transmeta already provides 16 levels. Secondly, we ignore memory operations for memory operation time in MPEG decoding is relatively small compared to the CPU operation time [13]. Thirdly, we exclude the display function of MPEG players because the state-of-the-art technology executes those routines separately on other hardware devices. The fourth assumption is that the machine was considered to use no energy when idle, which is a bit optimistic because a chip will draw a small amount of power while in standby mode. The fifth assumption is that it takes no time to switch speeds. In practice, raising the speed will require a delay to wait for the voltage to rise first, although we speculate that the delay is on the order of 10s of instructions. Finally, we assume that the voltage is exactly proportional to the operating frequency. This is a reasonable assumption except when the voltage approaches the threshold voltage. The detailed results may be distorted by these assumptions, but overall inclination of the effects of our algorithms cannot be cast away as all these assumptions are already proven to reasonable through extensive research on DVS.

Table 1: Sample video clips used in the experimental study

Movies	Avg GOP Size	Frames Per Second	Avg Dec Time of a frame (sec)	Dev. of GOP Dec Time (sec)	Fluctuation
Alien	96066	23.976	0.378	0.00920	0.0000745
Conan	76778	30	0.232	0.00057	0.000007
Shave	96065	23.976	0.531	0.00115	0.000025
Xmen	76775	30	0.272	0.00029	0.000001
Hollow	96066	23.976	0.313	0.00094	0.000013
Angel	92131	25	0.267	0.00834	0.0000565

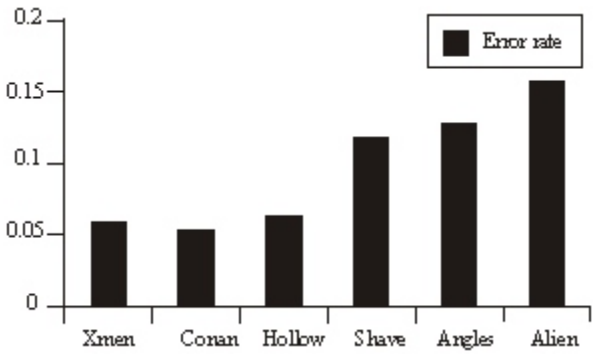


Fig2: Error rates of Prediction
(Streams are ordered with ascending degree of fluctuation.)

Simulation Results: In this section, we present the experiment results. We first show the accuracy of MPEG DT predictor and compare energy consumptions and real-time conformity of different algorithms. We measured error rates of DT prediction as Fig. 2. The graph shows that the accuracy of prediction is related with fluctuation.

Real-time conformities are shown in Fig. 3. The graph shows average number of frames that are missed or overly played at every checkpoint for real-time playback. The conformity of original MPEG player is not displayed as the conformity of both the DDVS and original is same: they all uses delay value as sleep time of next period. We can observe that PDVS outperforms especially in the streams that have high fluctuation such as *angels* and *alien*. In case of original MPEG player and DDVS, high fluctuation may delude determining *delay value* resulting in degraded QoS, whereas PDVS can adapt to a new workload environment through accurate prediction producing enhanced video quality. With these results, we can conclude PDVS algorithm improves real-time conformity of video playback by predicted MPEG DT.

Fig. 4 show the normalized energy consumption of decoders. Both DDVS and PDVS consume significantly less energy than shutdown mechanism does. It is also observed that in case of the shave, which has the longest average DT, there is relatively little energy reduction by both DDVS and PDVS. It is obvious that the longer the DT, the lesser room for energy reduction. The difference of energy efficiency between DDVS and PDVS is negligible.

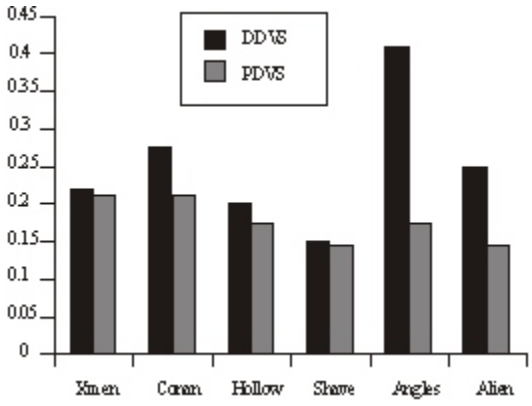


Fig 3: Average number of frames that are missed or over played at every checkpoint
(Streams are ordered with ascending degree of fluctuation.)

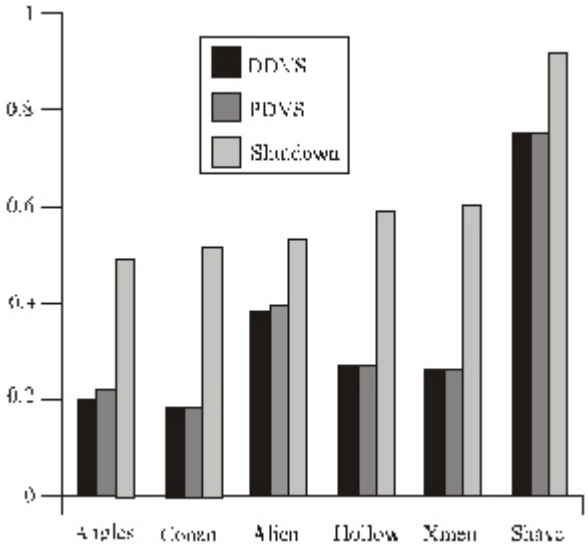


Fig 4: Normalized energy consumption
(Original MPEG decoder consumes one unit of energy)

CONCLUSIONS

Dynamic voltage scaling is a powerful methodology for reducing power consumption of a processor. But as it requires performance tradeoff, it should be applied carefully with information of future workload, which is the reason why application directed DVS is important. MPEG player is one of the most adequate examples for application directed DVS because its workload varies much which causes conventional DVS harms the real-time conformity, and future workload can be predicted. Our goal is to implement DVS for power efficiency while improving real-time feature of MPEG playback. We implemented two DVS algorithms for MPEG real-time playback: DDVS and PDVS. DDVS scales frequency and voltage only by previous workload while PDVS uses both previous workload and predicted MPEG DT. According to our experiment, power consumption of both DDVS and PDVS reduced significantly. In addition, the QoS of video play is enhanced with PDVS which is powered by DT predictor. We can also find that PDVS outperforms if the fluctuation of a stream high, which means the QoS of a video playback using PDVS is not as degraded by stream fluctuation as conventional MPEG players.

REFERENCES

Bavier, A., B. Montz, and L. Perterson,1998. Predicting MPEG Execution Times, SIGMETRICS/ PERFORMANCE '98, Int'l Conf. on Measurement and Modeling of Computer Systems, pp: 131-140

Boucher, J.A., Z. Yaar, E.J. Rubin, J.D. Palmer and T.D. C. Little,1995. Design and Performance of a Multi-Stream MPEG-1 System Layer Encoder/Player, MCL Technical Report 01-07- 1995, IS&T/SPIE Symp. on Electronic Image Science & Technology, San Jose, CA,

Burd,. T. and R. Brodersen, 1995. Energy Efficient CMOS Microprocessor Design, 28th Hawaii, Int'l Conf. on System Science, 1: 288-297.

Choi, K., K. Dantu, W. Cheng, and M. Pedram, 2002. Frame-Based Dynamic Voltage and Frequency Scaling for a MPEG Decoder, Int'l Conference on Computer Aided Design, pp: 732-737

Chung, E., L. Benini, G. Micheli, 2002. Contents Provider-Assisted Dynamic Voltage Scaling for Low Energy Multimedia Applications, 2002 Int'l Symposium On Low Power Electronics and Design

Krunz, M. and S. Tripathi,1997. On the characterization of VBR MPEG streams, ACM SIGMETRICS, Int'l Conf. on Measurement and Modeling of Computer Systems, pp: 92-292

Mitchell, J. and W. Pennebaker,1996. MPEG Video Compression Standard, Chapman & Hall.

Patel, K., B. Smith and L. Rowe, 1993. Performance of a Software MPEG Video Decoder, First ACM Int'l Conf. on Multimedia, pp: 75-82.

Pering, T., T. Burd and R. Brodersen, 1998. The Simulation and Evaluation of Dynamic Voltage Scaling Algorithms, Int'l Symp. on Low Power Electronics and Design, LongRun technology from Transmeta. http://www.transmeta.com/crusoe/ lowpower/longrun.html

Pouwelse, J., K. Langendoen, R. Lagendijk, and H. Sips, 2001. Power-Aware Video Decoding, 22th Picture Coding Symposium, Seoul Korea

Rajalingam, A. and Dr.S.K. Srivatsa, 2008. Processor Power Consumption using DVS Scheduling, National Conference on Emerging Techniques in Communication Engineering, pp: 273-277

Rajalingam, A. and N. Kumar, 2009. Low Power Multimedia applications using buffers for DVS Algorithm. National Conference on Emerging Trends in computing and Information Technology, pp: 28

Weiser, M., B. Welch, A. Demers, and S. Shenker,1994. Scheduling for Reduced CPU Energy, First Symp. on Operating System Design and Implementation OSDI 94: 13-23