Reprinted from the

Proceedings of the Linux Symposium

July 23rd–26th, 2008
Ottawa, Ontario
Canada

Conference Organizers

Andrew J. Hutton, *Steamballoon, Inc., Linux Symposium, Thin Lines Mountaineering*
C. Craig Ross, *Linux Symposium*

Review Committee

Andrew J. Hutton, *Steamballoon, Inc., Linux Symposium, Thin Lines Mountaineering*
Dirk Hohndel, *Intel*
Gerrit Huizenga, *IBM*
Dave Jones, *Red Hat, Inc.*
Matthew Wilson, *rPath*
C. Craig Ross, *Linux Symposium*

Proceedings Formatting Team

John W. Lockhart, *Red Hat, Inc.*
Gurhan Ozen, *Red Hat, Inc.*
Eugene Teo, *Red Hat, Inc.*
Kyle McMartin, *Red Hat, Inc.*
Jake Edge, *LWN.net*
Robyn Bergeron
Dave Boutcher, *IBM*
Mats Wichmann, *Intel*

Low Power MPEG4 Player

Joo-Young Hwang
Software Labs
Samsung Electronics Co. Ltd.
jooyoung.hwang@samsung.com

Sang-Bum Suh
Software Labs
Samsung Electronics Co. Ltd.
sbuk.suh@samsung.com

Woo-Bok Yi
Software Labs
Samsung Electronics Co. Ltd.
woobok.yi@samsung.com

Jun-Hee Kim
Seoul National University
goldlion@davinci.snu.ac.kr

Ji-Hong Kim
Seoul National University
jihong@davinci.snu.ac.kr

Abstract

In this paper, design and implementation of a dynamic power management for a MPEG-4 player is described. We designed two dynamic voltage scaling algorithms (feedback-based and buffering-based) to adapt CPU voltage dynamically according to the variable bit rate of a movie. We experimented the algorithms with the open-source XviD player. Our modified XviD player can save up to about 50% power without performance degradation. We describe practical lessons learned in optimization of the algorithms.

Power management (PM) is an important issue for battery-powered Linux devices, particularly for video playing devices. Our work shows how a conventional open-source video player can be modified to save power significantly on a CPU with dynamic voltage scaling capability.

1 Introduction

Power consumption is one of the big issues facing mobile embedded devices. Multimedia playing is one of the key functions of recent mobile devices, and the multimedia player is running for most of the time while using those devices. Power management for efficient multimedia playing in mobile devices is an important issue.

There have been numerous research papers published on power management for server systems [10], soft-real-time systems [6], and up to real-time systems [12, 1, 11]. Some of them are focused on per-component power management or system-level power management. In this paper, we focus on power management of the CPU component for multimedia playing devices, which is a soft-real-time system.

Many modern CPUs provides dynamic changing of clock frequencies and voltages in order to reduce power consumption. When a CPU becomes idle, it normally enters idle mode, a low-power mode provided by most current processors. However, it is rather efficient to reduce voltage levels as much as possible without violating deadlines of tasks, because power consumption of digital CMOS circuits is quadratically proportional to the supply voltages ($P = \alpha C f {V_{DD}}^2$)[2, 7].

Conventional DVS algorithms adjust CPU voltage/clock frequency dynamically according to workload variations. There are two DVS (Dynamic Voltage Scaling) approaches to exploit CPU slack time: inter-task and intra-task. Inter-task DVS is to exploit the slack time obtained from one task when the task completes prior to its planned execution time for the next task to schedule. The OS scheduler determines the CPU voltage level for the task to schedule, and changes the CPU voltage as determined.

Intra-task DVS is to change voltage levels during the execution of a task. Power Control Points (PCP) are inserted into a conventional program in order to change CPU voltage level at those points. PCP can be inserted automatically by the compiler, or manually by program-

mers. PCP should be inserted appropriately to avoid unnecessary frequent voltage switches, which will lead to performance degradation. There is no general rule of thumb on where to insert PCP into a program, but it may varies from application to application. The workload of an application should be analyzed statically or profiled dynamically at run time in order to pick the proper position for PCP.

Dynamic voltage scaling for multimedia players is an intra-task DVS method. There have been many papers on dynamic voltage scaling multimedia playback [3, 4, 8, 5, 9, 6]. In this paper, we describe a case study on practical issues in deployment of DVS algorithms on Linux and legacy multimedia player. We describe implementation issues of two DVS methods (feedback-based and buffer-based) and show detailed performance analysis. We also indicate possible further optimization directions.

This paper is organized as following: In Section 2, we overview currently known DVS algorithms and describe our implementation of those DVS algorithms in the open source XviD MPEG4 player, and discuss implementation issues in Section 3. Experimental performance results are shown in Section 4, and we summarize this paper in Section 5.

2 DVS Algorithms for Multimedia Playing

2.1 Feedback-based DVS Algorithm

Using conventional feedback-based algorithms [3, 4, 8, 5, 9], workload of the current Video Object Plane (VOP) is predicted considering the previous VOP decoding workloads, and the CPU voltage/clock is adjusted accordingly. In our media player, the VOP workload is estimated based on a simple moving average scheme; the workloads measured in the unit of the number of CPU cycles for previous frames are averaged over a window (typically 3 frames). Target CPU clock for the current frame decoding is calculated as the estimated workload divided by the decoding period. Then we select the power state with the minimum CPU clock among the CPU clocks higher than the calculated target frequency. The algorithm uses a different time window size for different VOP types (I, P, or B) of a frame.

2.2 Profile-based DVS Algorithm

This algorithm is the ideal case of a feedback-based algorithm. By profiling actual VOP decoding times in off-line and using the profile at run-time, the feedback-based algorithm can adjust voltage/clock according to correctly predicted workloads for all the frames. The VOP decoding times are measured for each CPU clock frequency because decoding time is affected not only by CPU clock frequency, but also by off-chip memory latencies. The performance of the profile-based feedback scheme gives the ideal performance which can be obtained by feedback-based algorithms.

2.3 Buffer-based DVS Algorithm

This algorithm, originally proposed in [6], can be used for input buffering or output buffering. The paper described only the input buffering case in detail, but we are interested in output buffering in this paper.

When there is no output buffer and the decoder should output decoded stream to the frame buffer directly, the decoder should wait until the next period to update the frame buffer. VST (Workload Variation Slack Time), which is generated by early completion of decoding of a frame, cannot be exploited for decoding of the next frame. If there are output buffers to save the decoded stream, the decoder can begin decoding of the next frame, exploiting the VST generated by the previous frame. This is illustrated in Figure 1.

When decoding of the j-th VOP is complete prior to its deadline, the slack whose amount is VST_j occurs. Without output buffers, the decoder should wait for this slack interval. Assuming that the workload required for decoding the $j+1$-th frame is PEC, the CPU clock frequency for decoding the frame is PEC/Period. With output buffers, the decoder can start decoding of $j+1$-th frame without waiting for the next period. At the moment, CPU clock frequency is adjusted to be PEC / (VST_j + Period), so power consumption is reduced. At the deadline of $j+1$-th frame, the current frame buffer address is switched to the memory, which contains the decoded data of the $j+1$-th frame.

According to [6], maximum buffer size can be estimated by the ratio of worst case execution time (WCET) to best case execution time (BCET). The VST increases as the actual execution time becomes shorter than the

deadline. Assuming the steady state worst-case scenario where VST is saturated to a maximum value, (BCET / WCET) becomes equivalent to (T / T + VST) where T is the period, then VST = T(WCET/BCET - 1). To fully exploit the VST, output buffers should be available. Therefore, the buffer size h should satisfy the following. $h \geq \lceil \frac{VST}{T} \rceil = \lceil \frac{WCET}{BCET} - 1 \rceil$

To determine BCET and WCET for a media clip, the clip is played once at the full speed of a CPU. The authors of [6] supposed that the execution times of the applications follow a normal distribution and took 3σ variations around the mean of the distribution as boundary values. In our experiment, we simply take the actual maximum and minimum execution times as WCET and BCET, respectively.

3 Implementation

3.1 Voltage Scaling Function in Linux Kernel

In our experimental platform, CPU supply voltage is regulated via the LTC1663 chip, and it takes non-negligible time to change CPU voltage. So, the multimedia player should not block waiting for completion of CPU voltage change. A voltage scaling daemon, running as a kernel thread, is responsible for changing CPU voltage to a new value, specified by the multimedia player's voltage scaling request. The VS daemon writes the request to the LTC1663 chip and sleeps on interrupt from the chip, which arrives when the actual voltage change is complete.

3.2 Media Player SW Architecture

Our low power media player consists of four modules as described in the following.

- AVI I/O Library

  Most MPEG-4 video data are contained in a separate container format such as AVI or MOV. Among them, AVI format is generally used, and we use it for our experiments. AVI I/O library extracts MPEG-4 VOP stream data from AVI file and sends it to XviD decoder. We use Transcode AVI I/O library for this module.

- XviD Decoder Library

  This is a open source GPL licensed MPEG-4 decoder library. It is not official reference software for MPEG-4, but it is fully compatible with MPEG-4 and its performance is well optimized, so we choose it for our baseline media player.

- Low Power Player Module

  This is the core module of our media player. It performs basic hardware initialization and sends frame data decoded by the XviD decoder library to the LCD frame buffer, in synchronization with frame deadlines. This also includes implementation of feedback-based and buffer-based DVS algorithms.

- User-level VS Library

  The VS library provides user-level DVS APIs for applications. These invoke system calls to get service of DVS functions from Linux kernel.

DVS APIs provided by the user-level VS library to the media player are summarized in the following.

- `unsigned int getCurrentSpeed()`

  returns current relative speed

- `void setScaledClock(unsigned int newSpeed)`

  set CPU clock frequency to a value corresponding to a given relative speed value of "newSpeed"

- `void setScaledSpeed (unsigned int newSpeed)`

  set both CPU clock frequency and voltage at the same time corresponding to a given relative speed value of "newSpeed"

- `unsigned int getCurrentVoltage()`

  returns current voltage value multiplied by 100.

- `void setVoltage(unsigned int newVoltage)`

  set CPU voltage to the given "newVoltage" which is a voltage value multiplied by 100.

3.3 Power States Selection

PXA-255 provides various CPU voltage/clock combinations, as shown in Table 1. We selected combinations with the same memory clock frequency of 99.5 MHz to avoid ambiguity, as described in the following. When we consider two combinations with different memory clocks and different CPU clocks, it is not deterministic which one consumes more power and gives higher performance, because the actual power consumption depends on workload characteristics. For memory intensive workloads, a combination with a higher memory clock may give higher performance, even though it has lower CPU clock frequency. It is also difficult to compare the total power consumption including CPU and memory power, because they may vary from device to device, and an accurate power consumption specification for processor and memory is required for correct comparison. For implementation simplicity and portability, we selected the four combinations in Table 1.

PXA-255 processor has an idle mode, where power consumption is minimal. It is generally entered by operating systems when the system is idle. When an interrupt arrives, processor mode is immediately changed to active mode. Even though the system is idle, the Linux kernel still typically processes a timer tick every ten milliseconds. To reduce the power consumption for processing, the periodic timer ticks when the system is idle; we adjust the CPU voltage/clock to the lowest power state on entry to slack interval.

3.4 Deadline Misses Handling

In the feedback-based method, a deadline miss may occur when the workload prediction is incorrect. If the CPU voltage is adjusted too low, the decoding of a frame will not be complete until the deadline of the frame. When such deadline miss occurs for a frame, the VOP deadline of the next frame is shortened accordingly not to increase the total play time.

4 Performance results

Our experimental platform, TynuxBox-Xe, is equipped with an Intel PXA255 CPU operating at 400MHz, 32MB SDRAM, and 32MB NOR flash memory. To measure the power consumption of the CPU, a data acquisition instrument is used to collect voltage samplings. Small serial resistance is inserted between the supply and the CPU to measure the current flow. Voltage level is sampled at the points (A) and (B) in Figure 2. Instantaneous power consumption of the CPU is calculated as $V_B \frac{V_A - V_B}{R}$, where V_A and V_B are the voltages at (A) and (B) points, respectively. Voltages are sampled at 1000 HZ frequency. We used a trailer for the Matrix Revolutions as an input, in which length is 63 seconds, video frame rate is 12 frame per second, and screen resolution is 240x160.

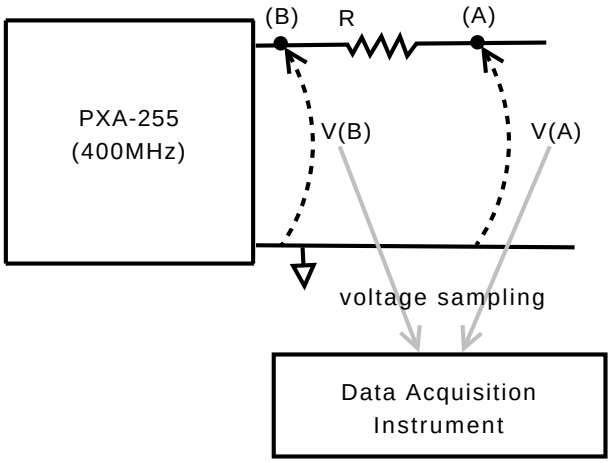
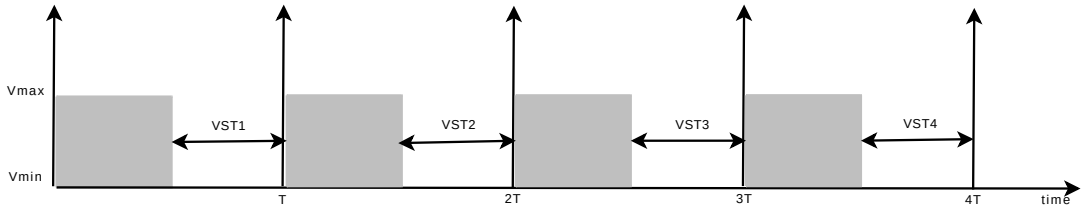


Figure 2: Power consumption measurement

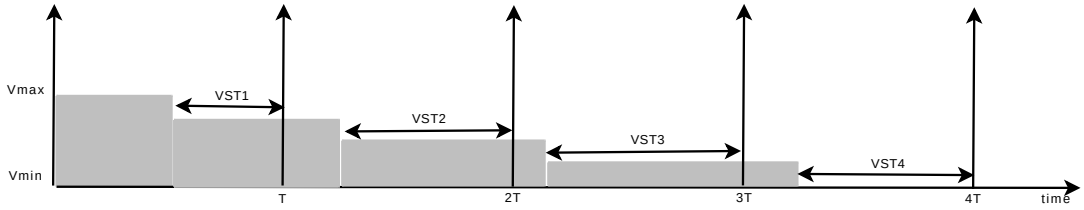
We compare the following four cases:

- Normal case without DVS methods. CPU voltage is set to the highest value when system is active, and CPU enters idle mode when system is idle.
- Feedback-based method. It is to adjust CPU voltage frame by frame according to the predicted current frame's CPU workload, which is estimated based on history of actual workloads of the previous frames. Moving average window size is 3.
- Profile-based method. It is to adjust CPU voltage frame by frame according to the actual workload which is known a priori.
- Buffer-based method. It is to adjust CPU voltage frame by frame according to the worst-case execution time of each frame. The number of buffers is set to 4, which is calculated as described in 2.3.

Figure 3 shows instantaneous power consumption of the three DVS methods for the first 500 milliseconds time



(a) Without output buffers



(b) With output buffers

Figure 1: Working behaviour comparison between w/o buffer and w/ buffer cases.

CPU Clock (MHz) corresponding to PXA255's CCCR setting (N)				
N = 1.00	N = 1.50	N = 2.00	N = 3.00	SDRAM Clk(MHz)
99.5@1.0V	-	199.1@1.0V	298.6@1.1V	**99.5**
132.7@1.0V	-	-	-	66
199.1@1.0V	**298.6@1.1V**	**398.1@1.3V**	-	**99.5**
265.4@1.1V	-	-	-	66
331.8@1.3V	-	-	-	83
398.1@1.3V	-	-	-	99.5

Table 1: PXA255 CPU voltage/clock combinations table with SDRAM clock frequency. CCCR is Core Clock Configuration Register of PXA255 CPU.

interval, during which the video changes smoothly and decoding workload is low. As shown in Figure 3 (a), the decoder in the normal case is active only for approximately half of the period. For the workload, all those DVS methods work well and none of them outperforms the others. Total energy consumption over the time interval are 11.49, 7.37, 6.23, and 7.26 mJ for normal, feedback-based, profile-based, and buffer-based methods, respectively. It should be noted that the buffer-based method adjusts the CPU voltage higher than the profile-based method for the first frame, because it uses the worst-case execution time for workload estimation, while the profile-based method uses the exact workload of the frame. For that reason, total energy consumption of the profile-based method is the the lowest among the three methods.

Figure 4 shows instantaneous power consumption of the DVS methods for the time interval from 3 - 3.5 seconds of the video during which the scene changes quickly, and decoding workload is high. The feedback-based method failed to reduce power for this workload because the moving average-based workload estimation is wrong for most cases. The buffer-based method works well for the workload, and even better than the profile-based method. In the method, CPU voltages are kept low for most of time and media player hardly enters Idle mode, which is owing to the efficient exploitation of VST. Unlike other methods, in the buffer-based method, CPU voltage change is not synchronized with frame deadline as observed during the (i+4)-th frame in Figure 4 (c). Energy consumptions during the time interval are 14.84, 14.91, 10.28, and 7.33 mJ for normal, feedback-based, profile-based, and buffer-based methods, respectively.

The current buffer-based method has more optimization opportunities. It is possible that CPU voltage is conservatively set because the buffer-based method uses the worst-case execution time of a video clip, instead of using feedback from actual execution times. In case of playing a video whose frame decoding complexity variation is very high, setting the CPU voltage considering WCET may lead to buffer shortage while decoding low complexity frames. Using actual execution times can be a solution of this problem. Since this may also cause deadline misses as the feedback-based method, either appropriate deadline miss handling or deadline miss avoidance is necessary, which is one of our future works.

5 Summary

In this paper, we described our implementation of DVS algorithms designed for low-power multimedia playback. Feedback-based and buffer-based methods are implemented using the XviD MPEG4 player. The feedback-based method performs well for smoothly changing video. However, it could not correctly predict frame decoding workloads for quickly changing video, which led to non-significant power reduction. We also implemented the profile-based DVS method, which uses correct workload information profiled at off-line to show the upper bounds of the performance, which can be obtained by ideal feedback-based methods.

The buffer-based method using multiple output buffers performs better than the profile-based method, owing to efficient exploitation of VST (Workload Variation Slack Time). Without output buffers, the decoder cannot decode the next frame, even though the current frame decoding is completed earlier than its deadline. In contrast, with output buffers, the decoder can continue work by queuing output to buffers. The number of buffers does not have to be large, and 4 buffers were enough to get significant power reduction for our test video sequence.

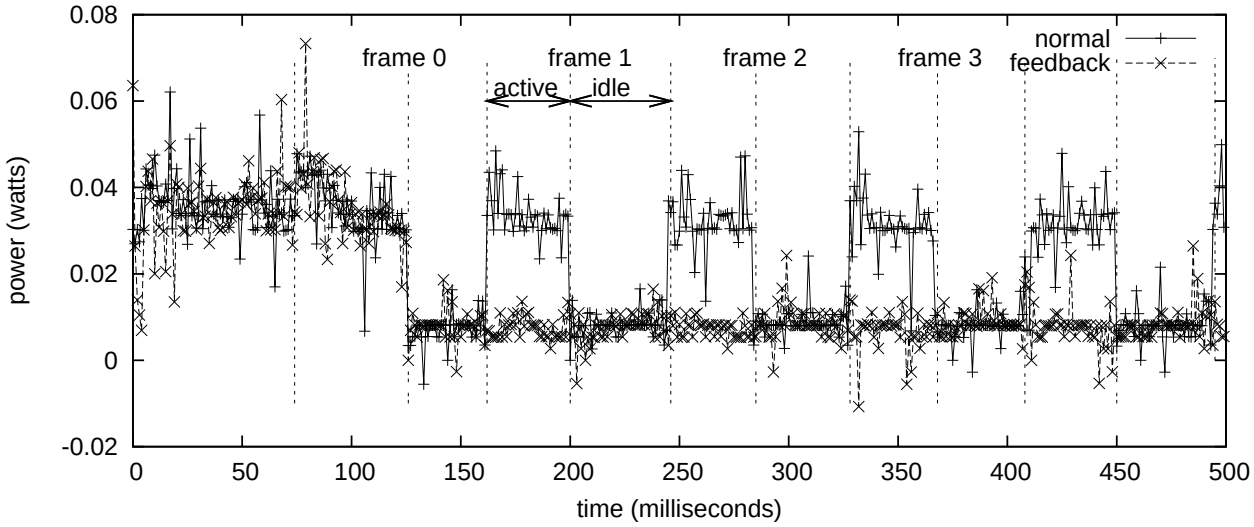
We showed detailed analysis of voltage scaling behaviour for typical DVS methods. We also indicate the possibility of further optimization of the buffer-based DVS method for handling video sequences with varying frame complexity.

References

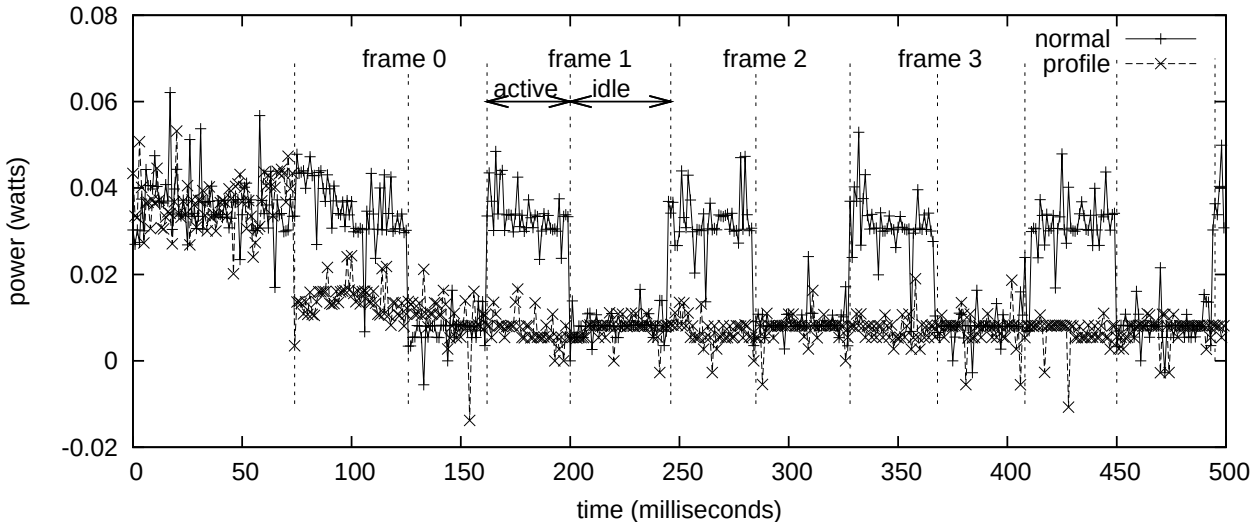
[1] H. Aydin, R. Melhem, D. Mosse, and P. M. Alvarez. Dynamic and aggressive scheduling techniques for power-aware real-time systems. In *Proceedings of IEEE Real-Time Systems Symposium*, 2001.

[2] T. Burd and R. Brodersen. Processor design for portable systems. In *Journal of VLSI Signal Processing*, Aug. 1996.

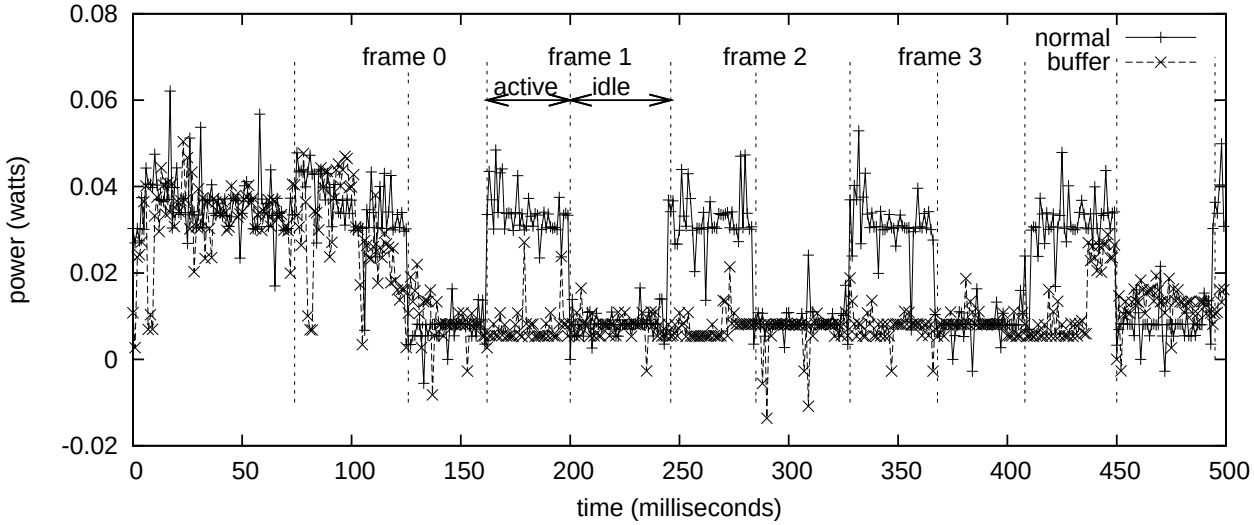
[3] K. Choi, K. Dantu, W. Cheng, and M. Pedram. Frame-based dynamic voltage and frequency scaling for a mpeg decoder. In *Proceedings of International Conference on Computer Aided Design*, pages 732–737, November 2002.



(a) Normal vs. Feedback based method

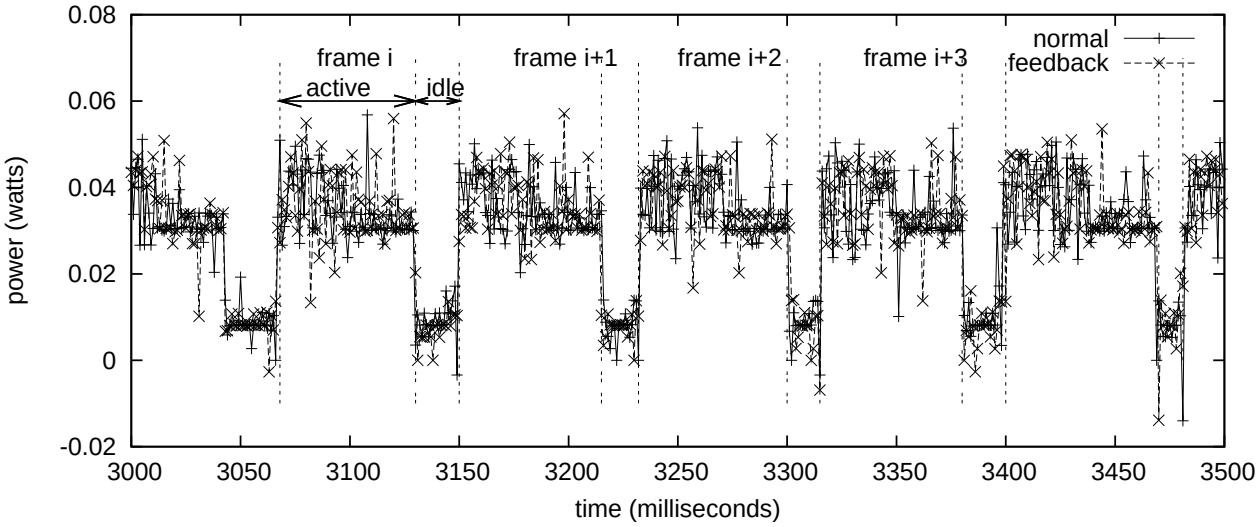


(b) Normal vs. Profile based method

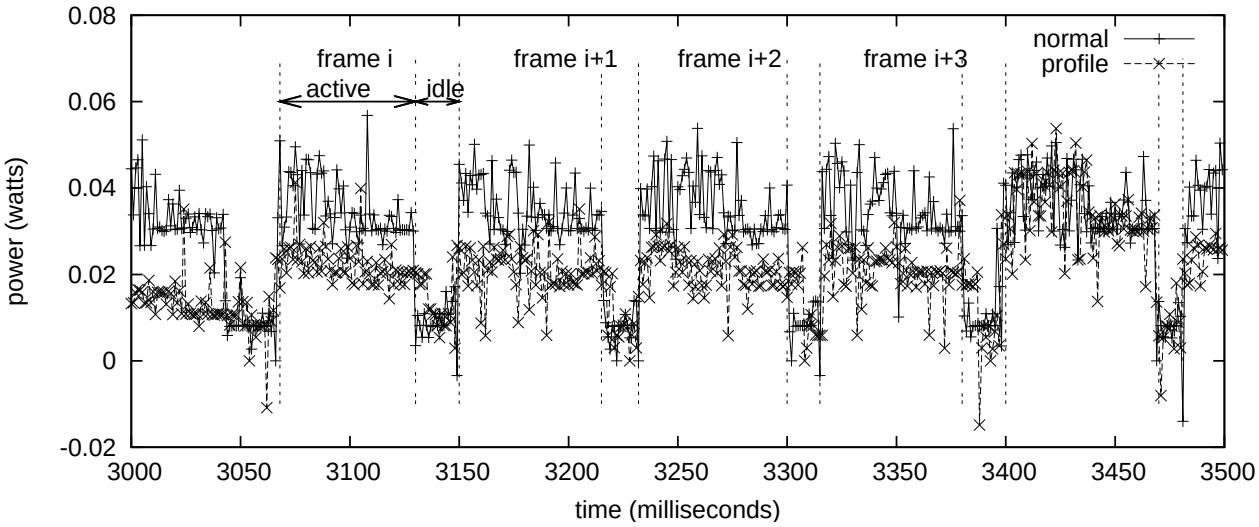


(c) Normal vs. Buffer based method

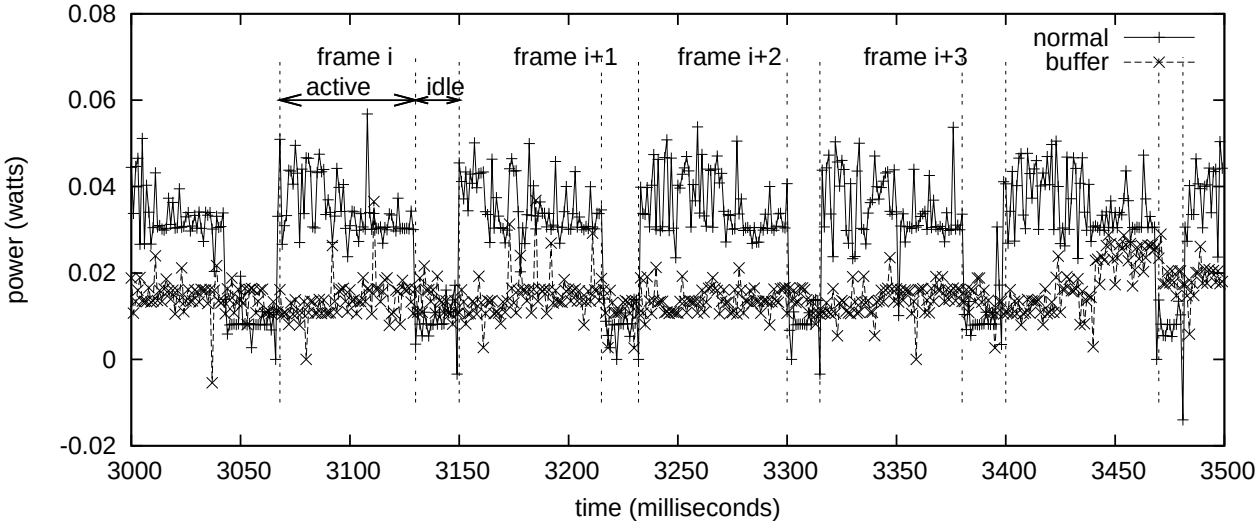
Figure 3: DVS results for smoothly changing video part.



(a) Normal vs. Feedback based method



(b) Normal vs. Profile based method



(c) Buffer based method vs. Normal and Profile-based Methods

Figure 4: DVS results for quickly changing video part.

[4] K. Choi, R. Soma, and M. Pedram. Off-chip latency-driven dynamic voltage and frequency scaling for an mpeg decoding. In *Proceedings of41st Design Automation Conference*, pages 544–549, June 2004.

[5] 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.

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

[7] T. Ishihara and H Yasuura. Voltage scheduling problem for dynamically variable voltage processors. In *Proceedings of ISLPED (International Symposium on Low Power Electronics and Design)*, Aug. 1998.

[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 489–496, October 2003.

[9] Z. Lu, J. Lach, M. Stan, and K. Skadron. Design and implementation of an energy efficient multimedia playback system. In *Signals, Systems and Computers, 2006. ACSSC '06. Fortieth Asilomar Conference on*, pages 1491–1497, Oct.-Nov. 2006.

[10] T. Pering, T. Burd, and R. Broderson. The simulation and evaluation of dynamic voltage scaling algorithms. In *Proc. Int'l Symp. on Low Power Electronics and Design*, pages 76–81, 1998.

[11] P.Pillai and K. G. Shin. Real-time dynamic voltage scaling for low-power embedded operating systems. In *Proceedings of 18th ACM Symposium on Operating Systems Principles*, pages 89–102, 2001.

[12] Y. Shin, K. Choi, and T. Sakurai. Power optimization of real-time embedded systems on variable speed processors. In *Proceedings of the International Conference on Computer-Aided Design*, pages 365–368, 2000.