Power Reduction via Macroblock Prioritization for Power Aware H.264 Video Applications

Michael A. Baker, Viswesh Parameswaran, Karam S. Chatha, and Baoxin Li
Department of Computer Science and Engineering
Arizona State University
Tempe, Arizona 85281
{mike.baker, vparames, karam.chatha, baoxin.li}@asu.edu

ABSTRACT

As the importance of multimedia applications in hand-held devices increases, the computational strain and corresponding demand for energy in such devices continues to grow. Portable multimedia devices with inherently limited energy supplies face tight energy constraints and require optimization for energy conservation. Power-aware applications give their users flexibility to prioritize and trade between performance and battery-life.

This paper introduces a power-aware technique for user selectable power reduction in exchange for controlled reductions in video quality for H.264 video streams. The technique uses an encoder-decoder pair. The encoder characterizes video streams and provides information to the decoder via Flexible Macroblock Ordering (FMO) by generating prioritized slice groups. The decoder selectively ignores low priority slice groups based on user selected preference effectively reducing the decoder workload. With a reduced computational requirement, processor voltage and frequency scaling (DVFS) significantly improve decoder power performance within timing constraints. Our PXA270 system implementation resulted in power savings of as much as 53% with an average PSNR per frame of 24dB compared to the unmodified video.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: [measurement techniques, performance attributes]; K.8.1 [**Personal Computing**]: Application Packages—*graphics*

General Terms

Performance, Measurement

Keywords

H.264, MPEG4, Video, Low Power, Power Aware, Voltage Scaling, Frequency Scaling

1. INTRODUCTION

Multimedia applications are among the most computationally expensive commonly found applications in portable devices. They are also among the most power hungry. Dedicated portable multimedia devices such as MP3 players are increasingly expected to deliver video content as well as the traditional audio. Cell phones and handheld computers have begun to merge into a new class of devices facing the same demands. Consumers find devices which can provide all of these services from telephone to personal digital assistant to internet browsing and watching video-on-demand more convenient than having the same services provided by multiple devices. A portable device struggling with a myriad of tasks and a limited power budget faces numerous opportunities for power conservation by focusing effort and limiting resources on certain tasks. The concept of Power Aware devices as presented in [12] asserts that the device should be able to choose an appropriate power saving mode by direct user input, based on the user's history, or automatically by sensing the environment.

Consider a situation where a portable multimedia device user wants to watch a video program for the duration of airline flight, but remaining battery power only affords video for half the duration. Under these circumstances users are willing to sacrifice Quality of Service (QoS) in exchange for increasing the time service is available. Power Aware applications provide users the flexibility to exercise this type of trade-off.

Our Power Aware H.264 system provides user selectable degrees of power saving effort in exchange for *controlled QoS reduction.* Our goal is to reduce the amount of computation required to decode the H.264 stream. We achieve this goal by skipping carefully selected blocks during video stream decoding. Based on the number of skipped blocks, we can predict the resulting speed-up in the decoder and use Dynamic Voltage and Frequency Scaling (DVFS) to reduce power consumption. Our H.264 encoder-decoder system classifies macroblocks in each frame into slice groups using Flexible Macroblock Ordering (FMO) described in the H.264 standard [15]. Our prioritization algorithm determines which blocks in each frame are the most expendable and organizes them into slices accordingly. Slice groups corresponding to different QoS measures are then selectively omitted from the decoding process based on user preference. We compare the effect on QoS and power savings using two types of error concealment. Simple *Copy Forward Error Concealment* (CF-ERC) provides replacement data from the previous frame with a very inexpensive copy based solely on macroblock location. *Motion Vector Error Concealment* (MV-ERC) is more computationally expensive, but provides substantially improved QoS.

The experimental results in Section 5 demonstrate that both techniques are capable of enabling significant power savings of up to 53% and 29% compared to the fully decoded video stream. In some cases, particularly using Motion Vector error concealment, the power savings are significant, while degradation to QoS is minimal. Significantly, our encoded stream is also compatible with standard H.264 decoders without support for our slice dropping scheme.

Next we present some relevant background on H.264 and QoS concepts. Previous work will be discussed in Section 2, in Section 3 we introduce the encoder-decoder system, and discuss modifications to the JM version 12.4 reference encoder and decoder [2]. Our experimental setup is explained in Section 4 followed by our results and conclusion.

1.1 H.264

The H.264/Advanced Video Coding (AVC) codec is part of the MPEG-4 suite of multimedia standards. The first revision of the standard, completed in May 2003, sought to address increasing demand on video bandwidth associated with growing services like High Definition Television and internet based streaming video applications [15].

H.264 implements a number of improvements over previous video codecs which collectively result in reductions of up to 50% in bandwidth requirements for the same level of image quality. The savings result from extensive analysis and optimization in the encoder which serves to minimize redundancy in the video stream, but comes at a significant cost in computational complexity. The H.264 decoder's complexity is about 2.4 times that of a comparable H.263/MPEG2 decoder [4].

Reduced bandwidth makes H.264 bitstreams particularly useful for hand-held applications using video streams transmitted over a wireless network where bitrates are limited and bandwidth comes at an absolute premium. In addition, storage capacity in a mobile device may also be limited. Unfortunately the increased computational complexity that comes with reduced bandwidth has obvious negative consequences for portable devices relying on battery power supplies.

1.2 Quality of Service

For our purposes, QoS is objectively defined as the level of distortion introduced into a video at the individual frame level and across several frames as a result of discarding residual data from macroblocks. We use Peak Signal to Noise Ratio (PSNR) calculated between the unmodified video and the same video with discarded data. We consider a PSNR value around 25dB to be acceptable for a low power mode. Values close to 35dB and higher are considered acceptable for more expensive computations. This range corresponds with the expected PSNR performance of a lower quality low bit-rate video stream vs. a high bit-rate [3]. More details are provided in Section 3.3. The Video Quality Measurement Tool from MSU is used to collect PSNR data for individual video frames [13].

2. PREVIOUS WORK

Several publications have focused on the effects of network packet loss on streaming video quality, methods for recovering from packet loss and encoder optimizations designed to improve video streams' robustness against packet loss. In [11], a priority drop scheme manages prioritized data flow on a network for frame dropping, and dynamically variable coefficient quantization enabling graceful QoS degradation under variable network conditions. Our encoded stream is suited for similar packet prioritization, but without necessary modifications to the video coding standard. Additionally, reducing quantization levels in the decoder does not provide significant computational savings relative to block or frame dropping, and slice dropping provides better resolution version of frame dropping. [6] analyzes the video stream in the compressed domain, pruning the compressed data based on a model which estimates the effect on distortion. This scheme is aimed at mobile devices on a network where each device has limited computational ability considered by the video server which appropriately modifies the compressed data stream. The transcoding scheme focuses on computational constraints while minimizing the introduced distortion, but it relies heavily on frame dropping and does not provide measured distortion injection in contrast to our approach.

Video decoding workload is highly variable. Significant effort has gone toward predicting workload for the purpose of DVFS for power optimization. [14] gives two methods for predicting workload in MPEG streams in order to effectively scale frequency and voltage. [3] describes a method for dynamically choosing IDCT algorithms in order to exchange quality for energy. In [5] three techniques are introduced for taking advantage of multimedia applications tolerant to deadline misses enabling opportunities for DVFS. Here, data points are dropped from the stream exchanging quality for power savings; however, unlike our scheme, the impact of individual data points on QoS is not considered.

3. POWER AWARE H.264 APPLICATION

We implemented our encoder-decoder pair by modifying the JM H.264/AVC reference encoder and decoder version 12.4. Our goal was to free the decoder from the maximum amount of computation while minimizing the degradation introduced into the decoded video stream. We accomplish this by selectively skipping or dropping from the video stream blocks which introduce the least amount of distortion into the video output. The decoder need not perform dequantization or inverse integer transform operations on the dropped blocks, effectively reducing the time required to decode a given video stream. The resulting speedup enables application of Frequency and Voltage Scaling for potentially significant reductions in power consumption.

Our encoder generates H.264 video streams inserting an I frame every twelfth frame from which no blocks will be dropped followed by eleven P frames with prioritized blocks. We use the Group of Pictures (GOP) sequence:

$$\{I, P, P, P, P, P, P, P, P, P, P, P\}$$

The I frame uses intra-coding exclusively so that no error introduced during block dropping will propagate beyond this frame. The P frames in the sequence use inter-coding in which blocks from each P frame may be reconstructed using data from the previous I or P frame in the video sequence. These inter-coded frames will propagate errors from one frame to the next when a motion vector points to areas in the previous frame impacted by dropped blocks.

At this point we avoid introducing B frames which use motion vectors in both the forward and reverse direction into the data stream in order to simplify analysis of the decoded video stream. However, adding B frames to the encoded stream in the future will improve bandwidth efficiency, and reduce the propagation of distortion introduced by dropped macroblocks. Since B frames are not normally used as reference frames for Copy Forward or Motion Vector Copy operations, blocks dropped from B frames will not propagate at all. Additionally, inserting a B frame between two P frames means that the effect of distortion spreading to multiple blocks through multiple motion vectors is limited to one stage over three frames instead of two as is the case with a sequence of three P frames. Adding B frames to the stream does; however, increase the computational load on the decoder, potentially reducing power efficiency.

3.1 Error Concealment

H.264 includes provisions for error concealment to minimize video quality degradation in the event of lost data such as dropped network packets or missed decoding deadlines. Error concealment functions typically replace an entire missing frame or block with buffered data from a previously decoded frame. In the JM reference software, "conceal by copy" replaces a missing block with its predecessor from the previous frame. A "conceal by trial" replaces the missing macroblock after evaluating the surrounding blocks to find one whose motion vector points to data minimizing the distortion at the edge of the missing block. These two actions loosely correspond to the error concealment actions taken by our two decoder schemes. We use "conceal by copy" for error correction in the simple CF-ERC decoder, and a modified version of "conceal by trial" for the MV-ERC decoder. Since we selectively drop available blocks in the decoder, any motion vector data calculated for a given macroblock is still available to us. This provides the opportunity to replace the macroblock using its motion vector data rather than a simple copy forward. This method is guaranteed to introduce the minimum amount of distortion into the video stream as the unmodified H.264 motion compensation algorithm in the encoder has carefully selected the motion vector data for each block to do exactly that.

3.2 Error Propagation

Error introduced in one video frame decreases as it propagates forward due to leakage in the prediction loop [9]. Although distortion introduced through macroblock dropping need only be considered over the course of a window of several frames [9], errors for a given frame can be approximated as the sum of the propagation errors since the last I frame [10].

Errors introduced through macroblock dropping propagate to the next frame each time a macroblock in the following frame references the dropped block via motion vector. All or part of the distortion caused by the dropped macroblock is carried forward when each dependent block in the following frame is reconstructed from its correct residual data added to the distorted data present in the previous (reference) frame.

We define the amount of error or *distortion* introduced into the video stream by a single macroblock, MB, in terms of Mean Squared Error (MSE). For our purpose, MSE defined between the nth block in the current frame m, MB_n^m and the block in the same location from the previous frame, MB_n^{m-1} is given by

$$\begin{aligned} &MSE(MB_n^m) \\ &= \tfrac{1}{16^2}\textstyle\sum_{i=0}^{15}\sum_{j=0}^{15}||MB_n^m(i,j) - MB_n^{m-1}(i,j)||^2 \end{aligned} \tag{1}$$

where $A(i,j)$ is the Y component of the pixel at position (i,j) in the 16×16 pixel block A. We derive the frame distortion from macroblock dropping, D_m from the sum of $MSE(MB_n^m)$ values across all n macroblocks in the frame

$$D_m = \sum_{i=0}^{n-1} MSE(MB_i^m) \tag{2}$$

The total distortion of frame m, $\hat{D}_m$ can be estimated as the sum of the distortion due to dropped macroblocks added to the distortion already present and propagated forward from the previous frame

$$\hat{D}_m = D_m + \hat{D}_{m-1} \tag{3}$$

Since our system does not drop macroblocks from I frames, and I frames are decoded independently without inter-coding, the distortion in any I frame is considered to be zero. Thus, the total distortion, $\hat{D}$ present in the first P frame, m following an I frame is $\hat{D}_m = D_m$. We can adjust for the attenuation of distortion from the previous frame, $\hat{D}_m$ by multiplying by an empirically determined scaling factor $0 \le \alpha \le 1$ before adding D_m giving

$$\hat{D}_m = D_m + \alpha \cdot \hat{D}_{m-1} \tag{4}$$

3.3 Encoder Modification

The modified JM12.4 reference encoder prioritizes and orders blocks within each frame according the amount of distortion, $MSE(MB)$ we would introduce into the stream if the block were dropped by the decoder. Blocks with the smallest MSE are considered more expendable, and are placed at the beginning of the list. Blocks with larger MSE are considered more important and are placed at the end of the list.

After prioritizing the macroblocks, the encoder begins dividing the blocks into 6 slice groups numbered from 0 to 5. The most expendable blocks are placed in *slice group 5* which will be the first slice dropped in the decoder. The most important blocks are placed in *slice group 0* which is never dropped in the decoder. Each slice group has associated with it an acceptable level of total distortion, $\hat{D}_m$ which acts as a threshold when assigning macroblocks to slice groups. The threshold is estimated from user selected Y-component PSNR values for each slice group. PSNR is given by

$$PSNR = 10 \cdot \log_{10}\left(\frac{MAX^2}{MSE}\right) \tag{5}$$

where MAX is the maximum pixel value, in our case 255. Given a desired value for $PSNR$, we can calculate the associated MSE by

$$MSE = \frac{MAX^2}{10^{PSNR/10}} \tag{6}$$

Since this value is an average across all macroblocks in one frame, we multiply MSE by the number of macroblocks in the frame to find the allowable distortion or *distortion threshold* for slice group s, D_s. For CIF video, 352×288 gives us 22×18 blocks, or 396 macroblocks per frame, and $D_s = 396 \cdot MSE$.

The encoder builds slice groups for each frame starting at the low distortion end of the prioritized macroblock list and adding macroblocks to the lowest priority slice group as long as the sum of their distortions does not exceed the previously calculated distortion threshold for that slice group. The result is a slice group for each distortion threshold such that dropping any slice and all higher numbered (more expendable) slices results in a level of distortion described by the distortion threshold for the dropped slice. During this process, the encoder also checks each macroblock's distortion in all three components (YUV) against an individual macroblock threshold and immediately adds failing blocks to *slice 0*. This check prevents the encoder from adding blocks when plenty of room for additional distortion is available to an expendable slice group, but adding it would introduce obvious artifacts into the video stream.

3.4 Decoder Modification

We modified the JM12.4 reference decoder to drop slices from the modified H.264 stream in accordance with a user selected mode. The user chooses a desired level of QoS performance in order to reduce power consumption in the decoder by selecting the number of slice groups to keep in the stream. For example, placing the decoder in *6 slice mode* will not drop any slices. On the other hand, placing the decoder in *1 slice mode* will drop slice 1 and any higher numbered slices, decoding only the most important slice, slice 0. Based on the user selected mode, the decoder performs error concealment to replace data in the skipped slices. We implemented two decoders—one using macroblock CF-ERC and the other using MV-ERC as described in section 3.1.

The decoding loop treats I frames and slice groups numbered smaller than the mode number normally, reading and decoding each macroblock. Slice groups with identification numbers equal to or greater than the mode number are skipped. The CF-ERC decoder does not read or decode blocks in these slices so that they are treated as lost by the error concealment function. The error concealment function is modified to ensure that conceal by copy is the only method used for error concealment, avoiding the conceal by trial function to minimize computational complexity for performance reasons. The MV-ERC decoder does read each macroblock in a dropped slice in order to obtain the associated motion vector data, but the coefficient data is not read, and the block is never decoded. Each block from a dropped slice is marked as lost in order to trigger error concealment, and a modified version of the conceal by trial function is used to recover the motion vector data for each marked macroblock copying data from the previous (reference) frame in accordance with the motion vector information.

In our testbed, preliminary results indicated that using motion vectors for error concealment in all three video components (YUV) was so computationally expensive that it wiped out any savings achieved through avoiding integer transform and dequantization operations. The process of rebuilding blocks by performing motion compensation can take 55% of the decoder's effort [1]. As a result, we use motion vector error concealment only for the Y component, and simple copy forward to replace dropped UV data.

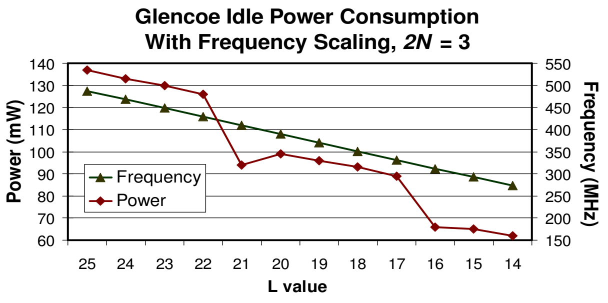


Figure 1: Glencoe DVFM Characteristics

Table 1: Speedup and DVFM frequency for decoder in 1-slice mode for CF and MV decoders.

	CF		MV	
Video	Speedup	f(MHz)	Speedup	f(MHz)
akiyo_cif	25%	351	9%	429
container_cif	57%	312	16%	409.5
highway_cif	43%	331.5	6%	448.5
soccer_cif	30%	370.5	5%	448.5
tempete_cif	65%	292.5	14%	429
waterfall_cif	57%	312	17%	409.5

The CF-ERC decoder provides better power performance than the MV-ERC alternative. In addition to skipping block decoding, it avoids the expenses of recovering motion vector information from the video stream and later dereferencing it to perform error concealment. However, the improved power performance implies reduced QoS. Although the encoder classifies blocks based on their distortion with respect to the same block (same position) used in CF-ERC, the MV-ERC method will regularly reconstruct a replacement block with less distortion resulting in improved QoS as discussed in Section 3.1.

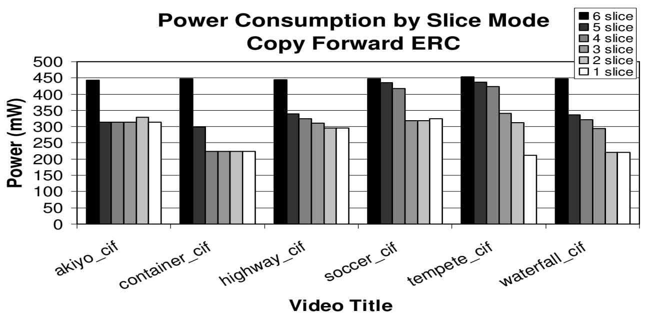
4. EXPERIMENTAL SETUP

We generated H.264 video streams for 16 benchmark videos using our modified JM12.4 encoder. These streams were then decoded through the two modified JM12.4 decoders on a Linux server, with an Intel(R) Xeon(TM) CPU running at 2.80GHz with 4GB RAM and on the Intel Glencoe Development Platform running Linux 2.6.9 on the XScale-PXA270 rev 4, with 64MB RAM.

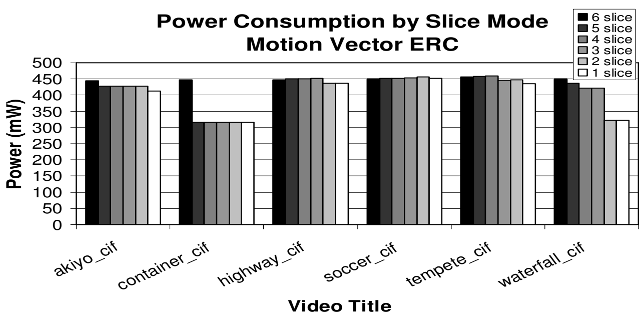
Power measurements were obtained for a subset of the 16 benchmark videos using the PXA270 V_{core} voltage touch point on the Glencoe Development Platform with a Keithley 2000 Multimeter sampling at 100Hz with the decoder running each video in each mode. Voltage samples were collected and analyzed in Labview to obtain an average power reading over several iterations of 1024 samples each. These results are presented in the next section.

The Linux port for the Glencoe Development board includes a utility for Dynamic Voltage and Frequency Management (DVFM) [7]. Frequencies are set by giving the utility a clock multiplier, L, and a turbo mode multiplier, *2N*. The base clock frequency of 13MHz is multiplied by $L \cdot N$ to obtain the desired frequency [8].

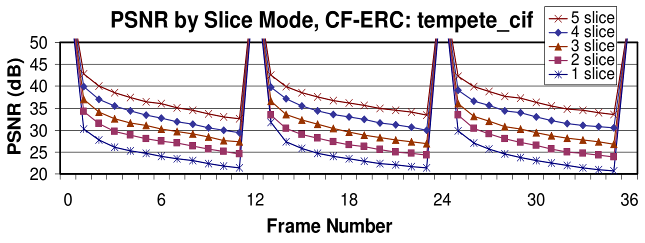
In order to obtain a piecewise-linear power function in terms of L and *2N*, it was necessary to fix the *2N* value.



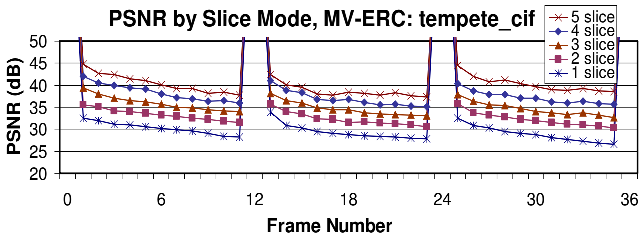
(a) Copy Forward Error Concealment



(b) Motion Vector Error Concealment



(c) Copy Forward Error Concealment



(d) Motion Vector Error Concealment

Figure 2: Power figures by slice dropping mode for Copy Forward (a), and Motion Vector ERC (b). The PSNR charts show QoS performance for each decoder mode on tempete_cif for Copy Forward (c) and Motion Vector ERC (d).

Table 2: Power and PSNR for decoder in 1-slice mode for CF and MV decoders.

	CF		MV	
Video	P	PSNR(dB)	P	PSNR(dB)
akiyo_cif	29%	43.0	7%	43.8
container_cif	50%	34.9	29%	37.6
highway_cif	33%	34.5	2%	35.7
soccer_cif	27%	28.7	0%	32.7
tempete_cif	53%	24.1	5%	30.1
waterfall_cif	51%	30.5	29%	33.8

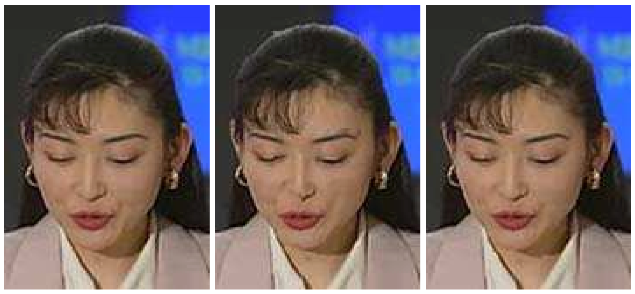
We chose a value of $2N = 3$ giving the suitable although non-ideal function of power in terms of frequency for system idle seen in Figure 1. We use frequencies ranging from the default value of 468Mhz to 273MHz indicated in the chart with L values from 24 down to 14. This range of frequencies crosses three available voltage levels as can be seen in the figure. The high step on the left corresponds to $V_{DD} = 1.5V$, the middle step corresponds to $V_{DD} = 1.4V$, and the low step on the right corresponds to $V_{DD} = 1.3V$. The power performance chart for the idle Glencoe board gives an idea of the power savings we should expect at various frequencies after implementing DVFM in the decoder. Reductions in V_{DD} provide significant power savings as expected from the quadratic relationship between power and supply voltage, $P = C_{switching} \cdot V^2 \cdot f$.

Our goal is to characterize the decoders' power performance given H.264 streams generated by the modified encoder. In the first step, we find the relative speedup experienced by the decoder when dropping slices and compare with the speed of the decoder decoding all slices. Relative speedup determines our ability to slow the decoder with frequency scaling in order to reduce power consumption. To collect decoder timing data, a 100 frame modified H.264 video stream was generated for each test video. Decoder frame rate data was then collected over several iterations for each video in each slice dropping mode on the Linux server and the Glencoe board in order to verify the Glencoe's performance.

In our analysis, we take the frame rate the Glencoe decoder achieves for the full six slice video as the nominal frame rate against which improved frame rates for slice dropping modes are compared for frequency scaling purposes. For instance, if the decoder in 1 slice mode decoding akiyo_cif finishes 25% earlier than the same video fully decoded, then we can slow the clock by 25% and still decode at the same frame rate. The adjusted frequency is given by $f' = t' \cdot f_{default}/t$ where t' is the improved decoding time, and $f_{default}$ is the default frequency 468MHz. This value must be adjusted up to the next available frequency in our DVFS scheme. For akiyo_cif, the calculated frequency, $f' = 348.7$MHz must be adjusted up to the next available frequency, 351MHz at $V_{DD} = 1.4V$.

5. RESULTS

Timing results for the CF-ERC and MV-ERC decoders in 1-slice mode are given in Table 1 with the computed DVFM frequency used for each video on the Glencoe board. The associated power savings for a given video and frequency are listed in Table 2 along with average P frame PSNR. Significant power savings were obtained from the CF-ERC decoder, in some cases without substantial impact on PSNR. The MV-ERC decoder achieves much smaller power savings due to the significant complexity of handling motion vectors


(a) akiyo_cif

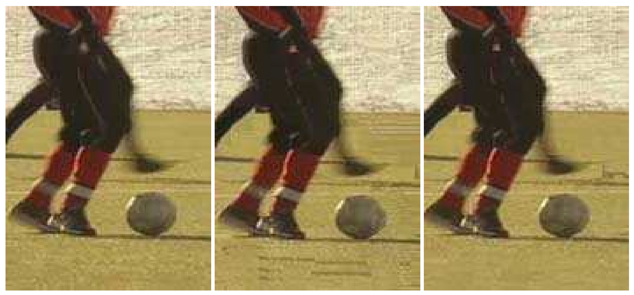

(b) soccer_cif

Figure 3: Example of last P frame decoded before next GOP demonstrating image quality for the Copy Forward (middle) and Motion Vector Error Concealment (right) vs. the fully decoded video (left).

during decoding, but it does a much better job of preserving image quality. The two videos with the lowest PSNR values decoded using the CF-ERC decoder see the largest QoS improvement when the MV-ERC decoder is used. An example of QoS performance of both encoders is presented over all slide dropping modes for tempete_cif in Figures 2(c) and 2(d). The controlled introduction of distortion into the video can be seen as well as the QoS improvement introduced through MV-ERC.

The power measurements obtained for each video and decoder slice dropping mode are shown for both decoders in Figures 2(a) and 2(b). The impact of voltage scaling is evident as power drops significantly each time reducing frequency makes a lower V_{DD} available as predicted in Figure 1. Figure 2(b) illustrates unpredictable performance when using MV-ERC. Performance varies for both CF and MV schemes with variations in the transmitted image. There may be significant variation in MB inter-dependence and motion-vector densities from one scene or even one frame to the next. The MV-ERC scheme is particularly susceptible to these variations due to the expense of decoding with motion estimation.

6. CONCLUSION

Our H.264 block prioritization scheme successfully enabled significant power savings in concert with DVFM. Our objective QoS measurements indicate acceptable quality performance, but there are several opportunities for improvement. The subjective quality of the decoded videos can be improved through further tuning of the parameters in the encoder to reduce artifacts in the decoded image, particularly focusing on distinguishing foreground objects from background. Tuning a number of other factors may also improve performance. Some examples are reducing the number of slice groups which decreases decoding complexity and bandwidth, considering motion vector data in addition to distortion data when prioritizing macroblocks, using motion vectors less aggressively in the decoder for MV-ERC, considering B frames, and varying the GOP size between I frames.

7. ACKNOWLEDGMENT

V. Parameswaran and B. Li are partially supported by ARO grant W911NF-06-1-0354.

8. REFERENCES

[1] Y.-K. Chen, E. Q. Li, X. Zhou, and S. Ge. Implementation of h.264 encoder and decoder on personal computers. *Journal of Visual Communication and Image Representation*, 17(2):509–532, April 2006.

[2] H.264/AVC. Reference software. http://iphome.hhi.de/suehring/tml/.

[3] R. Henning and C. Chakrabarti. A quality/energy tradeoff approach for idct computation in mpeg-2 video decoding. In *IEEE Workshop on Signal Processing Systems*. 2000.

[4] M. Horowitz, A. Joch, F. Kossentini, and A. Hallapuro. H.264/avc baseline profile decoder complexity analysis. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):704–716, July 2003.

[5] S. Hua, G. Qu, and S. Bhattacharyya. An energy reduction technique for multimedia application with tolerance to deadline misses. In *DAC Proceedings*, pages 131–136. June 2003.

[6] Y. Huang, A. V. Tran, and Y. Wang. A workload prediction model for decoding mpeg video and its application to workload-scalable transcoding. In *Proceedings of the 15th international conference on Multimedia*, pages 952–961. 2007.

[7] M. Ihmig. Porting linux 2.6.9 to the pxa270 based development platform. research collaboration between Intel and CMU, May 2005.

[8] Intel®. Intel®pxa27x processor family developer's manual. Intel Order Number: 280000-001, Apr. 2004.

[9] J. G. Kim, J. W. Kim, and C. C. J. Ku. Corruption model of loss propagation for relative prioritized packet video. In *SPIE Proceedings*. July 2000.

[10] J. S. Kim, J. G. Kim, K. O. Kang, and J. Kim. A distortion control scheme for allocating constant distortion in fd-cd video transcoder. In *IEEE International Conference on Multimedia and Expo*, pages 161–164. June 2004.

[11] C. Krasic, J. Walpole, and W. Feng. Quality-adaptive media streaming by priority drop. In *Proceedings of the 13th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 112–121. 2003.

[12] C. Lian, S. Chien, C. ping Lin, P. Tseng, and L. Chen. Power-aware multimedia: Concepts and design perspectives. *IEEE Circuits and Systems Magazine*, 7(2):26–34, Second Quarter 2007.

[13] MSU. Video quality measurement tool v. 1.52. http://compression.ru/video/.

[14] D. Son, C. Yu, and H.-N. Kim. Dynamic voltage scaling on mpeg decoding. In *Proceedings of the Eighth International Conference on Parallel and Distributed Systems*. June 2001.

[15] T. Wiegand, G. J. Sullivan, G. Bjntegaard, and A. Luthra. Overview of the h.264/avc video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):560–576, July 2003.