PROACTIVE ENERGY OPTIMIZATION ALGORITHMS FOR WAVELET-BASED VIDEO CODECS ON POWER-AWARE PROCESSORS

Venkatesh Akella Mihaela van der Schaar Wen-Fu Kao

Department of Electrical and Computer Engineering
University of California, Davis, CA-95616

ABSTRACT

We propose a systematic technique for characterizing *the workload of a video decoder at a given time and* transforming the shape *of the workload to optimize the utilization of a critical resource* without *compromising the distortion incurred in the process. We call our approach* proactive *resource management. We will illustrate our techniques by addressing the problem of minimizing the energy consumption during decoding a video sequence on a programmable processor that supports multiple voltages and frequencies. We evaluate two different heuristics for the underlying optimization problem that result in 50% to 92% improvements in energy savings compared to techniques that do not use dynamic adaptation.*

1. INTRODUCTION

Energy optimization of multimedia applications especially on battery operated devices such as laptops and PDAs is extremely important. Growing demand for higher data rates and enhanced functionality that results in more complex algorithms exacerbates this problem even further. Present day processors provide very sophisticated power optimization capability at the hardware level. For example, modern embedded processors like Intel PXA27x have sophisticated power management support such as normal, idle, deep idle, standby, sleep and deep sleep that have different power saving benefits and wake-up costs in terms of delay and power when transitioning from one state to another. Existing approaches to implementing video codecs on general-purpose processors are **oblivious** of such power management features and resort to a *worst-case design* philosophy. The objective of the proposed research is to develop an implementation framework that is not only **aware** of hardware specific details but also **proactively adapts** the implementation strategy to offer the best possible resource utilization. In recent years, some techniques have been proposed to optimize the energy of multimedia applications on programmable processors [2][3][4][6][7][8] but we will demonstrate shortly that they are mostly reactive in nature, whereby the scope of the benefits is limited. The distinction between reactive and proactive adaptation is as follows – in a proactive scheme the shape of the workload at a given time can be altered by the implementation, while in a reactive implementation the workload is not re-shaped, the system merely adjusts itself to the changing workload.

The paper is organized as follows. We define the problem more precisely in Section 2. Section 3 illustrates the proposed technique and the underlying algorithms. Results are described in Section 4 and conclusions and future work is discussed in Section 5.

2. DETAILED PROBLEM STATEMENT

The problems being addressed in this research can be illustrated with the following example. Consider a multimedia application that consists of decoding a sequence of data units (DU). The data units could have different granularities, i.e. it can be a video frame, a part of a video frame, a video packet, a collection of video frames, a group of pictures, etc. Based on the frame rate, there is a *worst-case* design parameter T that denotes the amount of time available for processing a DU. Depending on the time-varying characteristics of the multimedia content, the deployed compression algorithm and parameters and encoding/transmission bit-rate, not every DU needs the entire T to complete its execution. Often, the actual completion time of a DU is less than T.

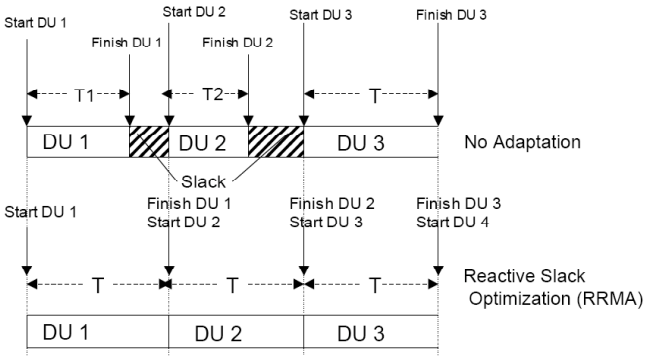

Figure 1 - Reactive Optimization

As shown in Figure 1, the first data unit DU1 needs T1 time units while the second data unit DU2 needs T2 time units. The difference between T and the actual completion time is called *slack*. This *slack* can be used to optimize resources such as energy in a system, by exploiting the fact that the system can be slowed down, so that it completes the task in *exactly* T units as shown in Figure 1 (bottom panel). The slowdown of the system can be accomplished by reducing the voltage or frequency or both for the system, which in turn reduces the energy consumed to complete the task. This is the key idea behind existing approaches to adaptive multimedia system design such as [2][3][4][6][7][8]. Clearly, this shows the *reactive* nature of the existing adaptation process i.e., predict the slack based on the time used by previous DUs and optimize the resources accordingly, i.e. react to the predicted slack. *The implementation does not alter the workload itself.*

0-7803-9332-5/05/$20.00 ©2005 IEEE

We propose a new scheme that is illustrated in Figure 2. As evident from the figure, the key idea of the proactive optimization strategy is to *change* the workload as seen by the processor. This is accomplished using two techniques. First instead of optimizing the slack for each DU, we propose to accumulate the slack over multiple DUs (as shown in the top panel of Figure 2) such that a more aggressive power savings mode can be used by the underlying processor– e.g.: deep sleep where most of the hardware units are shutdown, without incurring the associated wake-up penalty.

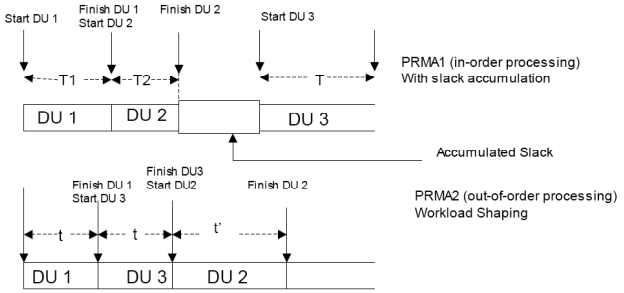

Figure 2. Proactive Optimization

We call this technique – PRMA1 (proactive resource management algorithm), and it processes the frames in the order in which they are received. A more aggressive form of workload shaping is shown in PRMA2 that re-orders the frames to select the best power saving strategy as shown in the bottom panel of Figure 2. This will increase the delay of some of the frames in the GOF (group of frames). This latency can be determined for different encoding parameters as shown in [12]. We will consider applications where such an increase in latency is acceptable such as video streaming. The key questions are when and how to accumulate and re-distribute slack so as to maximize the total power savings *without* compromising the quality i.e. for a given PSNR constraint. That is the central problem considered in this research.

Another difference between the techniques proposed in this paper and related work in literature such as [2,3,8] is that instead of developing on-line models for the complexity and *predicting* future complexity, we construct abstract complexity models at the encoder called *generic complexity metrics or GCMS* that are converted into real complexity metric (RCM) such as execution time, taking into account the specific architecture and resources available at a given decoder. The RCM is used to shape the workload as necessary. This is described in detail in our previous work [5,10]. The benefit of our approach is the ability to handle heterogeneous devices and avoid dropping frames due to misprediction errors.

3. METHODOLOGY

In this section, we describe the details of our research. First we show the system architecture and then present the details of the algorithms for workload shaping and proactive energy optimization. We use UMCTF based scalable wavelet video encoder [1] shown in Figure 3. Figure 4 shows the system architecture of the end-device. We show both the encoding and decoding paths. We introduce 3 new blocks – the GCM to RCM block, the complexity-modeling block and the resource manager block. We assume the resource manager has access to encoding parameters such as GOF (group of frames) size, temporal level t, bit-rate R, target frame rate f, and other encoding parameters. The main task of the resource manager is to construct a model for the workload in terms of the instruction count for the particular GOF and using the algorithms described below and set the configuration parameters for the underlying power management system and the task scheduler in the operation system.

3.1 Complexity Modeling

Execution time on a processor is equal to IC/IPC * T_c where IC is the dynamic instruction count, IPC is the instructions completed per clock cycle and T_c is the clock period of the processor.

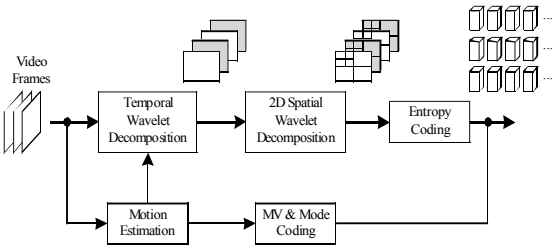

Figure 3. Block Diagram of UMCTF Based Wavelet Video Encoder

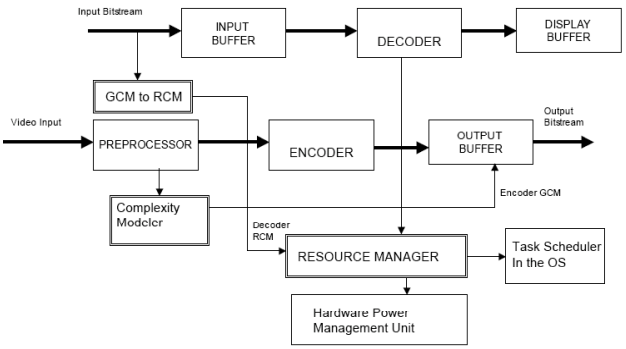

Figure 4 - System Architecture

Our profiling data shows that instruction count per clock cycle (IPC) of the same frame type roughly remains the same. Therefore, the instruction count (IC) is a good metric to measure the actual execution time complexity of a GOF. The complexity-modeling block generates the GCM as a function of the encoding parameters and the sequence characteristics as described in our previous work [5,10]. This is converted into actual instruction count (i.e. the RCM) by the target device taking into account the specific architectural features of the device such as memory bandwidth, instruction set, functional units etc. The RCM profile also referred to as the workload profile of a GOF of 32 frames is shown in Figure 5.

3.2 Resource Management Algorithms

Next, we describe how the resource manager uses the actual workload profile to optimize the energy consumed in decoding a GOF using the techniques illustrated in Figure 2. We evaluated **four** different algorithms on this system architecture: NOADP, RRMA, PRMA1, and PRMA2 that are described next.

1. NOADP (No Adaptation): This model simply decodes each frame at the highest speed of the underlying processor. If a frame is finished earlier, the processor is idle and the hardware still consumes energy. This is the baseline against which we compare the different algorithms.

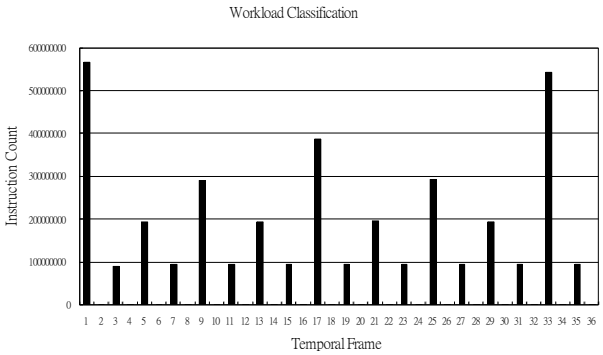

Figure 5 - Workload Profile for a GOF of 32 with 5 temporal levels

2. **RRMA (Reactive Resource Management Algorithm):** This illustrates the approach shown in Figure 1.

STEP1. Track the maximum instruction count of the past N frames of the same type. Use that as the estimate for the current workload along with a correction factor. This is similar to the model used in [6,8].
STEP2. Look up the configuration table using predicted instruction count. Select the frequency and voltage with lowest energy that can finish the workload within deadline. If none of them meets the goal, choose the highest available frequency.
STEP3. Change the CPU frequency and voltage and execute the frame. If a frame is finished earlier than the deadline, put the CPU to idle state until the deadline is reached.
STEP4. If the deadline is missed, modify the correction factor. Go to STEP1 until all frames are processed.

3. PRMA1 (Proactive Resource Management Algorithm (in-order): Please refer to Figure 2 for an illustration of this algorithm. PRMA1 is aware of the structure of decomposition tree and will redistribute the slack in a GOF. It requires a display buffer and a working buffer. PRMA1 estimates each frame's workload by converting the GCM to RCM as described in the previous section and processes each frame in temporal order. It uses a greedy algorithm and tries to use the lowest possible energy state to execute a frame but meets the deadline of a GOF, i.e. never misses a deadline. It is described below.

STEP1. Compute the actual workload IC_i for the frame i ($1 \le i \le$ P, where P is the number of frames in a GOF) as described in the previous section
STEP2. Determine the lowest possible energy consumption state S_i such for frame i that

$$\sum_{u=1}^{i-1} t_u + \frac{IC_i}{IPC * f_i} + \frac{IC_{GOF} - \sum_{u=1}^{i} IC_u}{IPC * f_{\max}} \le t_{GOF},$$

where t_u and IC_u are the actual execution time and instruction count, respectively, from frame u, IPC is the instruction count per clock cycle which is a constant, f_i is the frequency of operation corresponding to state S_i, f_{max} is the maximum frequency supported by the processor , t_{GOF} and IC_{GOF} are the total time to process the GOF and the total instruction count of the GOF.
STEP3. Set the processor to state S_i and process the frame.
STEP4. When a GOF is finished, put the CPU in the minimum energy state based on accumulated slack. Go to STEP1 until all frames are processed.

4. PRMA2 (Proactive Resource Management Algorithm (out-of-order): In the previous algorithm (PRMA1) we decode the frames in the order in which they were encoded and transmitted. To reduce the number of reconfigurations and maximize the energy savings, we decode the frames one temporal level at a time, starting with the lowest (most important) level. We denote this decoding mechanism "bottom-up". Note that in this approach we might not be able to finish the processing of all frames at the same frequency and voltage setting and yet meet the deadline. Hence, we will prioritize the frames (temporal subbands) in different classes, each class corresponding to a specific distortion impact and also frequency/voltage level. Initially, we can start our processing using a low voltage for the first t-1 temporal levels, and subsequently, we can adjust the frequency/voltage level to ensure that the frames in the GOF are entirely processed in the available amount of time allocated for that GOF. Our complexity-control algorithm is similar to the rate-control mechanisms available in the literature, with the important difference that for this we rely on complexity models and not on rate-distortion models to select a specific frequency/voltage operation mode.
This "bottom-up" frame-reordering algorithm has the important advantage that it reduces the fluctuations in the workload within a certain (short) time-window. The resultant workload is produced by PRMA2 is shown in Figure 6.

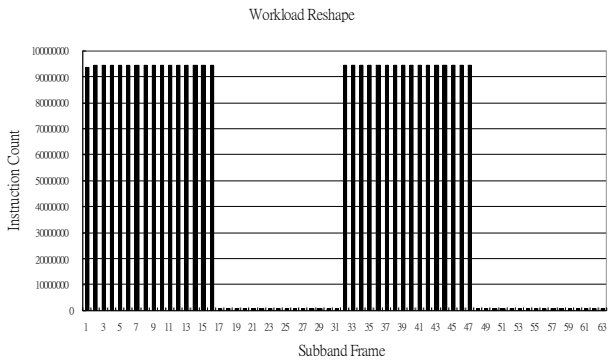

Figure 6 - Workload Reshaped by PRMA2

4. RESULTS

In this section we will present the experimental results and estimate the overhead introduced by the proposed algorithms.

4.1 Experimental Setup

We use Intel Pentium 4 Xeon 2.2 GHz running Fedora Core 2 Linux for our experiments. We used PIN tool [11] for obtaining the instruction count. We modified the SIV [9]

(Scalable Interactive Video) codec to measure decoding time for each frame. We assume the target architecture supports ten different configurations ranging from 700 MHz to 3.6 GHz with the corresponding voltages in the range 1.1 V to 1.5 V. We use the Akiyo and Foreman sequences in QCIF format for evaluating the energy saving. Both of them are encoded at multiple bit-rates. We profiled one GOF for each sequence to obtain the parameters we need. The two QCIF sequences encoded at 256Kbit/sec with five temporal levels. The frame rate is set to 24 fps, which corresponds to the processing rate supported by the Xeon processor at the maximum frequency and voltage for the highest data rate i.e. 256 kbps.

4.2 Results and Discussion

Table 1 - Comparison of Different Algorithms (r is rate, T is number of temporal levels)

	Akiyo QCIF r=256Kbps T=5	Foreman QCIF r=256Kbps T=5	Foreman QCIF r=256Kbps T=4	Foreman HQCIF r=256Kbps T=5
NOADP	0%	0%	0%	0%
RRMA	36.95%	35.61%	36.98%	80.21%
PRMA1	44.91%	44.43%	44.74%	88.47%
PRMA2	50.63%	50.32%	50.26%	85.48%

Table 1 compares the different algorithms with different sequences, different temporal levels and different frame sizes.

As expected, proactive strategies (PRMA1 and PRMA2) perform better than the reactive strategies (RRMA) and result in significant power savings ranging from 50% to 86% over a non-adaptive baseline codec. Note that the codec we use is not complexity scalable, so the energy saving is roughly the same for sequences with different motion characteristics (Akiyo and Foreman).

Table 2 - Comparison with different bit rates (r is rate, T is number of temporal levels)

	Foreman QCIF r=256Kbps T=5	Foreman QCIF r=128Kbps T=5	Foreman QCIF r=64Kbps T=5
NOADP	0%	0%	0%
RRMA	35.61%	63.12%	91.22%
PRMA1	44.43%	70.75%	92.23%
PRMA2	50.32%	73.72%	92.28%

Table 2 shows the impact of the bit-rate on the power savings. For lowest bit rate (64kbps)), all the algorithms use the same (optimal) power savings mode i.e. the lowest frequency and voltage on the processor since the workload is small. However, at high bit rates, the complexity is higher, and a clear advantage can be seen when using the proposed proactive strategies.

4.3 Overhead of Resource Management Algorithms

For RRMA algorithm, every frame introduces the following additional computational overhead - finding the maximum of N previous instruction counts, checking if the last frame misses the deadline or not, adding an adjustment factor to workload, and a table-lookup. PRMA1 requires two multiply and one add operation to estimate the base workload, two multiplications for the adjustment factor, one subtract, five multiplies, three divide and one compare operation for making the decision. The computational overhead is negligible on modern processors as it is in the order of hundreds of nanoseconds in the worst case. PRMA2 uses a working buffer and display buffer and this results in an increase in the memory requirements and latency.

5. CONCLUSIONS & FUTURE WORK

We proposed algorithms for proactive workload adaptation using high-level complexity models and a framework to interpret the models at run-time to choose the appropriate frequency and voltage of operation to minimize energy without loss in quality i.e. without missing any deadline for a frame. Furthermore, we illustrated the feasibility of the technique on the next generation wavelet based scalable video codecs. As noted before, the proactive algorithms require additional memory and also introduce a fixed additional latency, so the scheme may not be appropriate in applications that have strict delay constraints such as video-conferencing.

Acknowledgement: The authors would like to acknowledge NSF Award 0429154 and Intel Research for supporting this research.

6. REFERENCES

[1] J.R. Ohm, M. van der Schaar, J. Woods, "Inter-frame wavelet coding – Motion Picture Representation for Universal Scalability", Image Communications, Special issue on Digital Cinema, to appear June 2004.

[2] Daniel G Sachs, Sarita V Adve, Douglas L Jones, "Cross-layer Adaptive Video Coding To Reduce Energy on General Purpose Processors", ICIP 2003, pages 25-28

[3] R. Cornea, S. Mahapatra, N. Dutt, A. Nicolau, N. Venkatasubramanium. "Integrated Power Management for Video streaming to Handheld Devices" TR-03-19, Department of Computer Science, University of California, Irvine, 2003.

[4] Albonesi, D. H,, Balasubormonian, R. et. al " Dynamically Tuning Processor Resources with Adaptive Processing", IEEE Computer, December 2003, pages 49-58

[5] Mihaela van der Schaar, Deepak Turaga and Venkatesh Akella, "Rate-Distortion-Complexity Adaptive Video Compression and Streaming ", ICIP, 2003.

[6] C. J. Hughes, J. Srinivasan, and S. V. Adve. Saving energy with architectural and frequency adaptations for multimedia applications. In Proc. Of the 34th Annual International Symposium on Microarchitecture, December 2001.

[7] Y.H. Lee and C.M Krishna. Voltage-clock scaling Adaptive Scheduling Techniques for Low Power Hard Real-time Systems, IEEE Transactions on Computers, vol 52, number 12, pages 1586-1593, Dec 2003.

[8] D. G. Sachs, W. Yuan, C. J. Hughes, A. Harris, S V. Adve, D L. Jones, R H. Kravets, K. Nahrstedt. GRACE: A Hierarchical Adaptation Framework for Saving Energy. Computer Science, UIUCDCS2004-2409, February, 2004.

[9] D. Taubman. High performance scalable image compression with ebcot. IEEE Transacations on Image Processing, vol 9, no 7, july, 2000

[10] G. Landge, M. van der Schaar, and V. Akella. Complexity Metric Driven Energy Optimization Framework for Implementing MPEG-21 Scalable Video Decoders. IEEE ICASSP 2005.

[11] http://rogue.colorado.edu/Pin/

[12] D. Turaga, M. van der Schaar, Y. Andreopoulos, A. Munteanu and P.Schelkens, "Unconstrained motion compensated temporal filtering (UMCTF) for efficient and flexible interframe wavelet video coding," EURASIP Signal Processing: Image Communication 2004.