Adaptation of Video Encoding to Address Dynamic Thermal Management Effects

Ali Mirtar, Sujit Dey
Electrical and Computer Engineering Department
University of California, San Diego
San Diego, USA
{amirtar, sdey}@ucsd.edu

Anand Raghunathan
School of Electrical and Computer Engineering
Purdue University
West Lafayette, USA
raghunathan@purdue.edu

***Abstract*— Dynamic Thermal Management (DTM) acts as a necessary tool for safe operation of systems and increases their lifetime; however, application of DTM affects system performance, and can significantly impact the quality of results of complex real-time applications such as real-time video encoders. In this paper, we propose a dynamic adaptation algorithm that can be used with an H.264 encoder to change its complexity in real time, and hence adapt its computational needs to the dynamic changes in system performance due to DTM, while minimizing the impact on the quality and bit rate of the encoded video. We formulate our dynamic adaptation approach as a multi-dimensional optimization problem that maximizes video quality and minimizes bit rate while ensuring that the video encoder can run in real-time in spite of the DTM effects. We have implemented our adaptation algorithm with x264, a very efficient and commonly used H.264 encoder. We evaluated the adaptive encoder on a computing platform with Intel® 1.8GHz Core™2 Duo processor, which employs DTM based on DVFS. Our measurements with several video clips reveal that because of the dynamic effects of DTM, the quality of videos encoded by x264 are affected significantly - an average 10 dB reduction. However, with the aid of our dynamic adaptation algorithm, the x264 encoder can encode all the streams in real-time, with an average video quality degradation of only 2.4 dB, and with only a nominal increase in bit rate of the encoded streams.**

Keywords- Dynamic Thermal Management; Dynamic adaptation; real-time video encoding; H.264; Multimedia.

I. INTRODUCTION

Increased transistor scaling, along with rise in complexity and computational capabilities of today's semiconductors, have resulted in higher power density [1], and hence increased chip temperature [2]. High temperature reduces the lifetime of a chip in addition to the reduction in its functional reliability and performance, and increases its cooling cost [3]. Due to limitations of cooling such as high cost for servers and data centers, or space and/or power requirements in portable devices such as smartphones and laptops, dynamic thermal management (DTM) has been developed as a supplement and sometimes, an alternative solution to the conventional cooling for thermal management of semiconductors [3]. Several DTM methods have been developed for servers [4][5], general purpose processors [6], multiprocessors [7], mobile platforms [8] and embedded systems [9]. While these techniques are efficient in managing temperature, they can introduce different types of performance related overheads, which can lead to increasing the run time of the tasks/applications, and potentially impacting the quality of their results, in particular for real-time applications. In this paper, we will develop techniques to ensure that the negative impact of DTM on the performance and quality of the applications is minimized.

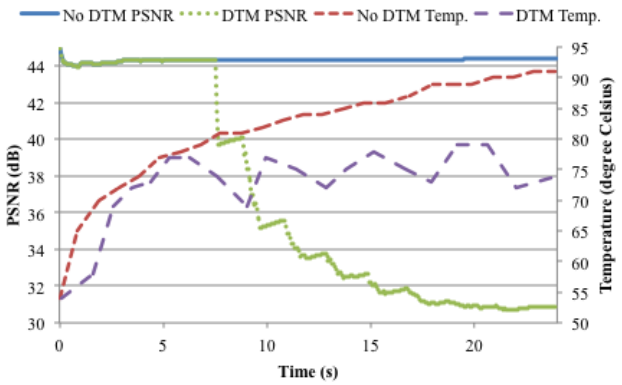

Figure 1. Effects of DTM on visual quality of a real-time video encoder.

According to several recent market research reports, video traffic will dominate all Internet traffic soon. For example, according to the Cisco Visual Networking Index [10], global Internet video traffic will be 50% of all Internet traffic by 2012, and the sum of all forms of video (TV, video on demand [VoD], Internet, and P2P) will be approximately 90 percent of global consumer traffic by 2015! The above indicate that video applications, including video encoding/decoding, will dominate the workload of servers, personal computers and mobile platforms. Hence, in this paper we focus on H.264 video encoder, the most widely used video encoding standard, as the focus application. We study the impact of DTM on H.264 video encoder, and develop a DTM aware dynamically adaptive video encoding technique, which will vastly reduce the impact.

978-1-4673-2154-9/12/$31.00 ©2012 IEEE

A. Effect of DTM on H.264 Video Quality

Real-time video encoding has very high computational needs while at the same time having hard deadlines (frames per second). Figure 1 shows the results of encoding a VGA video clip of 24 seconds with an x264 encoder (a very efficient implementation of H.264 encoder [11]), implemented on a MacBook laptop, with and without the use of a commercial DTM tool, Cool Book [12]. As shown in Figure 1, due to its high computation needs, running the x264 encoder without DTM will result in temperature rising above 90℃, while the encoded video quality is 44 dB PSNR (peak signal to noise ratio, the most widely used video quality measure). On the other hand, the use of DTM is able to maintain the temperature close its desired value of 75℃; however, the use of DTM reduces the performance of the hardware, resulting in the encoder not able to meet its hard deadlines, and consequently degrading the resulting encoded video very significantly (by about 13 dB in our experiment). The above experimental data show that while commercially available DTM techniques may be capable of managing the temperature of a system at desired levels, but their use is unacceptable from the system's applications perspective, in particular for real-time applications.

B. Our Approach

Our DTM aware adaptive video encoding approach is based on the following two key properties of video encoding application. First, there is a correlation between the amounts of computation needed, the bit-rate of the encoded video stream, and the quality of the resulting encoded video. Secondly, there is an inherent tolerance towards a nominal variation from the specification of both the desired bit-rate as well as the video quality. The above two properties allow us to design thermal aware adaptive video encoder such that the applications would be able to adapt to the dynamic changes in hardware performance due to dynamic thermal management without causing any perceivable degradation in user experience. We note that it is important to recognize and consider the multi-dimensional nature of the tradeoff between encoding speed, video quality, and bit rate. For example, it is possible to drastically improve encoding speed (while maintaining video quality) by encoding each frame as an I-frame (I-frames have been defined in section II), or by even skipping compression of frames; however, both these options would lead to unacceptable increases in the video bit rate.

In this paper, we develop a systematic approach to design a DTM aware real-time video encoder based on H.264 standard. To achieve this goal, we first study the parameters used in the encoder, and model their effects on different attributes of the video encoder, namely encoding speed, video quality and bit rate. We also characterize the impact on the application (encoder) speed for each allowed setting of the DTM, and produce a thermal policy characterization (TPC) table. Subsequently, we develop an application adaptation algorithm, which uses the identified parameters and their speed-quality-bitrate model, and the TPC table, to dynamically adapt the video encoding application in real-time in response to DTM induced encoder performance changes, such that the impact on the quality of the encoded video is minimized. Using the same experimental setup described in Section I.A, we demonstrate that use of the dynamic adaptation technique in conjunction with DTM can reduce the average drop in video quality to only 2.4 dB at a marginal 4% bitrate increase, as opposed to the 9.8 dB video quality loss when DTM is applied without the dynamic adaptation. We perform extensive experiments using different videos and different DTM profiles to demonstrate the ability of the proposed dynamic adaptation approach to mitigate the video quality loss that DTM otherwise results in.

C. Related Work

Adaptive bit rate encoding [13] and adaptive bit rate streaming [14] have become widely used techniques to encode and deliver video with optimal video bit rates to address changing network conditions. Every video encoder has the ability to adapt in real-time the encoding tools and parameters it uses to adjust to different video content, quality objective, and desired bit rate. Adaptive bit rate streaming allows a video stream to be encoded at different rates, so the client can dynamically select the optimal bit rate to be delivered depending on the network condition as indicated by the client's buffer condition. However, all the above techniques do not take into consideration, and cannot address, dynamic changes in system performance as a result of dynamic thermal management techniques, which is the problem we address in this paper.

Recently, adaptive video encoding techniques have been developed to enable scaling of encoding complexity to enable real-time encoding even on constrained computing platforms, or when encoder is implemented in software [15][16]. The main idea is to design a single video encoder application that is able to run in real-time on different platforms without any platform specific programming such as Assembly codes. The encoder regulates its complexity based on average encoding time by using faster but less optimal versions of various encoding steps including motion estimation. Adaptive techniques have also been developed to make video encoding energy aware, primarily in the motion estimation step [17]. These techniques, while changing the complexity of encoding, do not address the fast and dynamic changes in the performance of the same platform that dynamic thermal management produces, which is the focus of our work. The other difference between our work and these techniques is that we recognize and exploit the multi-dimensional nature of the tradeoff between encoding speed, video quality, and bit rate. Therefore, our adaptation methodology can not only maintain real-time behavior of the encoder, but also work under specified quality and bit rate constraints.

On the other hand, there has been recent effort on modifying or developing new thermal management techniques for multimedia applications such as video encoding or decoding [3][18-23]. In most of these work [3][18-21], existing DTM techniques have been modified to ensure real-time encoding/decoding, by making use of additional time available to process low complexity video frames. In [22], the authors develop a new thermal

978-1-4673-2154-9/12/$31.00 ©2012 IEEE

management approach based on balancing the encoding workload between different components of a video encoder in a multi-core processor platform. In [19], a new scheduling based DTM for MPEG-2 video decoding was proposed, based on the observation that there is a residual time after decoding every frame in MPEG-2 video decoding which can be utilized for thermal management.

The above approaches have several limitations. Firstly, they need modification of existing DTM techniques, or development of new ones. Secondly, these approaches are applicable on a specific DTM method or platform. For instance, [19-20] are based on dynamic voltage and frequency scaling only, [22] is based on multi-processor platform for load balancing. Thirdly, some of these techniques have very limited applicability; for example, a DTM method which works for MPEG-2 video decoding based on residual time per frame like [23] will not apply to more computationally intensive video use cases like video encoding, video compression standards like H.264, or even decoding of high frame rate videos.

In contrast, our approach focuses on adapting the application (encoding) to the dynamic changes induced by DTM, and hence it will work with existing DTM software, without the need to develop application specific DTM techniques and software. Moreover, even though due to implementation limitations we were able to test and report results using DVFS based DTM only, our approach can be used with a broad range of DTM methods (DVFS, multicore methods such as load balancing and deep-sleep modes, clock gating, fetch gating, etc.) because it is independent of the exact mechanisms employed by any specific DTM method but rather works with the performance impact produced by DTM. Thirdly, our adaptation algorithm does not need to make any assumption about computing or residual time of frames, and hence can be applicable to any video encoding standards, and videos with any spatial or temporal characteristics. And finally, we believe it is the first attempt to dynamically control video encoding complexity in response to dynamic thermal management, and perform multidimensional optimization of encoding speed, quality and bit rate.

The rest of paper is organized as follows. In section II, we first discuss a set of encoding parameters and describe the effects they have on encoding speed, quality, and bit rate. Next, we formulate DTM aware adaptation of video encoding as a multidimensional optimization problem, and describe the overall platform with interactions between the adaptive encoder, DTM controller, and the hardware. In Section III, we describe the adaptation algorithm in details. We provide the details of our test platform and the results in Section IV, and conclusions in Section V.

II. DTM Aware Adaptive H.264 Video Encoder

As stated in the previous section, our goal is to develop an adaptation technique associated with H.264 video encoding, which will be able to monitor the impact of a given DTM process in real-time, and adjust the encoding tasks dynamically so as to mitigate as much as possible the impact of DTM on video encoding quality. We start this section by discussing the relevant video encoding parameters that we will use for adaptation, and point out the tradeoff they can effect in encoding speed, video quality and bit rate. Next, we formulate the DTM effects aware video encoding adaptation as a multidimensional optimization problem, and finally introduce the overall adaptive platform which appropriate interactions between encoder adaptation, DTM controller, and the underlying hardware .

A. H.264 Video Encoding Parameters and Tradeoffs

Video encoders, in general, are lossy data compressors that reduce the size of a raw video sequence based on two factors, human's limited visual perception and redundant information in the video sequence. The influence of these two factors on video encoding can be tuned by some of the encoder parameters. The encoder parameters of interest are Quantization, Group of pictures, Number of reference frames and Search range. Selection of these parameters impacts the attributes of encoded video such as video quality, bit rate and also affects the speed of video encoding. In the following paragraphs, we introduce these parameters and briefly describe their effects on the encoder attributes.

Quantization is the main parameter that influences encoding based on human's limited visual perception. It is used by the encoder to drop out details of video sequences that are unlikely to be perceived by human eyes, thereby reducing the amount of encoding that needs to be done. Higher quantization results in dropping more details of video and less computation time. In addition, using higher quantization also significantly reduces the bit rate and the file size of encoded stream, thereby reducing the application time needed for memory access as well. However, higher speed and lower bit rate comes with the possible price of losing video quality.

Group of Pictures (GoP), Number of reference frames and Search range are the parameters that mainly influence the encoder's effort for finding redundant information in the video sequence. In H.264 standard, there are 3 different frame types I, P, B. When a frame does not require other video frames to decode and is encoded only with its own information it is called an I-frame. A P-frame can use its own data or the data from previous frames to encode which makes it more compressible than I-frames. Finally, B-frames are the ones that are encoded based on the information from both previous and forward frames and therefore they have the highest amount of data compression. The maximum effort for reusing redundant data occurs in B-frames and minimum efforts are done in I frame.

GoP size is the total number of frames starting from an I-frame and a sequence of P-frames and/or B-frames until the next I-frame. Higher GoP translates into better usage of redundant information in the video and reducing the final bit rate of video encoder. However, this comes with the price of higher computation and encoding time to find these information and some times even lower video quality. The maximum number of frames to search for redundant information and the range of search in each of them are set by Number of reference frames and Search ranges parameters. It is clear that using higher values for these

978-1-4673-2154-9/12/$31.00 ©2012 IEEE

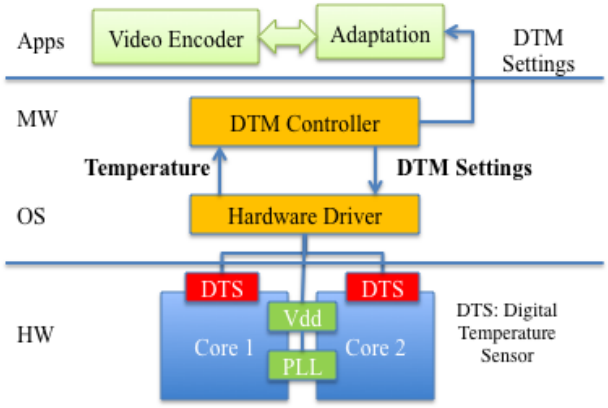

Figure 2. Adaptive H.264 Video Encoder Platform

parameters will result in lower bit rate and better video quality, but at the expense of more computation and run time.

B. Adaptation Problem Formulation

As can be seen from the discussion in the previous section, we can potentially use the four video encoding parameters to affect the computation complexity and hence speed of video encoding. Therefore, when video encoding performance is impacted by DTM, choosing the right parameter values can help reduce the computational needs of the encoder and still encode video sequences in real time. On the other hand, any such change in the values of the encoding parameters affects the bit rate and quality of the encoded frames. It should be the responsibility of the dynamic adaptation algorithm to address the performance impact due to DTM while keeping encoded video quality as high as possible and bit rate as low as possible.

We formulate our dynamic adaptation algorithm problem as follows. We consider 3 combinations for video encoder and their hardware platforms: first combination, E_o, is the original encoder on hardware where there is no change in the performance due to DTM. The second combination, E_n, is also based on original encoder but it is on a platform that incorporates DTM, which results in dynamic changes in the hardware performance. The third combination, E_a, uses adaptive video encoder and it is based on the same DTM platform. The attributes of each encoder, E_i, is defined as follows:

$Q(E_i)$: Video Quality of E_i in terms of PSNR.

$B(E_i)$: Bit rate of E_i in terms of kbps.

$S(E_i)$: Encoding speed of E_i in terms of frames per second.

Our proposed adaptation algorithm has to change the values of the encoding parameters to satisfy the following conditions:

$$S(E_a) \geq S(E_o)$$
$$Max\{Q(E_a)\}$$
$$Min\{B(E_a)\} \quad (1)$$

Note that since the impact of DTM on the speed of the encoder can vary during an encoding session, satisfying the above conditions will require the adaptation algorithm to be dynamic, in real-time adjusting the parameters to address the DTM impact.

While solving above optimization problem we expect the solutions to fall into the following boundaries:

$$Q(E_n) \leq Q(E_o) - \epsilon \leq Q(E_a), \quad \epsilon > 0$$
$$B(E_n) \leq B(E_a) \leq B(E_o) + \delta, \quad \delta > 0 \quad (2)$$

The first boundary means that the adaptive encoder not only should produce higher quality results than non-adaptive encoder with DTM but also its quality should not drop more than ϵ compared to ideal encoder. The second boundary says that it is also accepted to have bit rate more than ideal encoder but it should not exceed a limit of δ.

C. Interactions between DTM and Adaptive Encoder

Figure 2 shows the different components and layers of the overall system, with interactions between the DTM controller, hardware driver, and the underlying hardware on one hand, and the proposed adaptation layer on the other hand. The DTM Controller makes the decision for changing hardware settings based on the temperature it receives from the hardware driver. The hardware driver has the responsibility to read the temperature from Digital Temperature Sensors (DTS) in the hardware, and send appropriate commands to reconfigure hardware for thermal management purposes. When the DTM controller changes its settings, it conveys the information to both the hardware driver as well as the encoder adaptation layer. The adaptation algorithm checks in frequent intervals, and adapts the encoder parameters dynamically according to the algorithm described in the next section.

III. Adaptation Algorithm

In this section, we introduce our algorithm to dynamically adapt the encoding parameters in response to performance changes effected by DTM. The algorithm uses (a) an experiment-based model that associates values of the encoder parameters with values of the encoder attributes, and (b) a Thermal Policy Characterization (TPC) table that characterizes the effect of each possible DTM setting on the performance of the encoder. We first describe how we derive the Parameters-Attributes model, and then how we do the thermal policy characterization. Finally, we discuss the adaptation algorithm.

A. Parameters – Attributes Model

We find out the relationship between encoding parameters and attributes in the following way. We define a four-dimensional space in which each coordinate represents one of the parameters of the encoder. Therefore each point in this space such as $C = (q, g, r, s)$ corresponds to an encoding configuration with Quantization of q, GoP of g, Number of reference frames of r, and Search range of s. This four dimensional space can be defined by the following mathematical representation:

$$D = \{(q, g, r, s) |\ q \in [1,50],$$
$$g \in [1,250],$$
$$r \in [1,5],$$
$$s \in [4,16]\}$$

978-1-4673-2154-9/12/$31.00 ©2012 IEEE

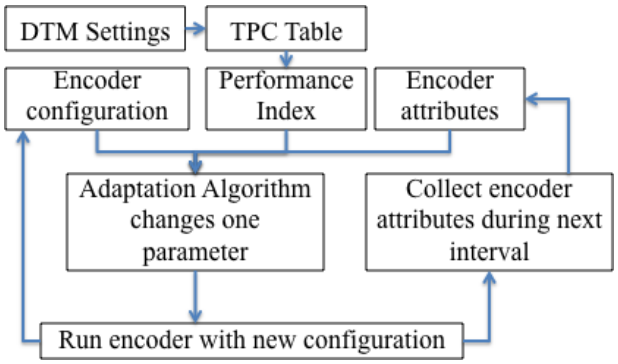

Figure 3. Workflow of Adaptation Algorithm

The domain range for each parameter is based on the actual encoder application limits.

For each encoding configuration point C, the encoder produces an encoded video with the set of attributes: visual quality in the form of Peak Signal to Noise Ratio (PSNR), encoded video bit rate (BR), and encoding speed in the form of frames per second (FPS). We define the Parameters–Attribute model as three functions PSNR(C), BR(C), and FPS(C). To derive the model, we performed experiments with a set of videos, encoding the videos with multiple encoding configurations and collecting the values of the resulting attributes. Since the four dimensional space D of parameter configurations is very large, and it would have taken months to perform encoding of the sample videos with all the possible encoding configurations, we have selected a subset of D as follows:

$$D = \{(q, g, r, s) | q \in \{18,23,28,33,38\}, \\ g \in \{1,2,4,8,16,32,96\}, \\ r \in \{1,2,3,4,5\}, \\ s \in \{4,8,12,16\}\}$$

For the rest of the points in D, we used linear approximation based on their adjacent points.

Please note that we do not use the value of attribute functions from sample video directly. These values will be used as a guide on how changing parameters will affect each attribute. For instance, if quantization changes from 23 to 28, how much it affects the encoder speed or quality. In section III.C, we show that the experimental model data will be normalize and scaled before being used for different videos.

Next, we describe characterization of DTM techniques impacts on system performance.

B. Thermal Policy Characterization

There are different methods for dynamic thermal management such as dynamic frequency scaling, dynamic voltage and frequency scaling, clock gating, load balancing, deep-sleep modes, etc. To be able to use our encoding adaptation algorithm with any of these different DTM methods, we define a unified variable among all DTMs called Performance Index that is calculated in the thermal policy characterization process.

We start from the fact that each of these DTM methods has a limited number of settings that it changes based on system temperature, and other factors like number of cores, number of voltage levels, etc. It means that at each moment, system is working at exactly one of its DTM settings. To characterize thermal policy, we first disable DTM in the sense that it does not dynamically switch between its different settings. Then for each setting we encode a series of video clips with the original video encoder and measure the total encoding time. For each setting we calculate the encoding speed, which is equal to the number of frames encoded divided by the total time of encoding:

$$Encoding\ Speed = \frac{Total\ Number\ of\ Frames\ Encoded}{Total\ Time\ of\ Encoding}$$

The encoding speed unit is frames per seconds (FPS). Therefore, for a DTM setting, S, the encoding speed is shown as FPS_S. Now we define the *Performance Index (PI)* of a setting, S, as follows:

$$PI_S = \frac{FPS_S}{FPS_N} \quad (3)$$

where FPS_N is the encoding speed of the nominal hardware setting with DTM turned off. It means the maximum value of PI is PI_N, which is equal to one. In fact, performance index indicates how much an application (here video encoding) is slowed down because of DTM settings change.

The values for each DTM setting and the corresponding PI are stored in a lookup table, termed the Thermal Policy Characterization (TPC) table. At any time, the adaptation algorithm can look up the TPC table to determine the current performance impact on the application due to DTM's current settings. Note that for a given system, the thermal policy characterization has to be performed with the DTM software used by the system only once, and offline. We will show in Section IV.A results of a characterization we have done for the MAC system we use in this paper.

C. Adaptation Algorithm

In response to changes in system temperature, DTM changes its settings, thereby changing the Performance Index. Given the current Performance Index, the objective of our adaptation algorithm is to select a point in the configuration space D, such that the video encoder will still work in real time in spite of the performance impact of the DTM, and will also produce maximum video quality and minimize bit rate considering the performance index change due to DTM.

Figure 3 shows the high-level workflow of the proposed adaptive encoder to reach this goal. As shown, the adaptation algorithm is called periodically after a time interval to incorporate the DTM effects. At the end of each time interval, the encoded video attributes collected during the interval, the new performance index resulting from the current DTM settings (from the TPC table), and the current encoder parameters are sent to the adaptation algorithm. The adaptation algorithm selects one of the parameters whose value is determined to be most effective to solve the optimization problem (Equations 1 and 2) for this time interval.

The reason for changing only one parameter in each time interval is twofold. First, calculating all the parameters at the same time can be very expensive in terms of computation needs (details of adaptation algorithm in the following

978-1-4673-2154-9/12/$31.00 ©2012 IEEE

paragraphs). Secondly, even if we select the best encoding configuration in each interval by changing all the parameters simultaneously, that configuration may not remain the best for the rest of the time interval, due to the possible dynamic change in both the video content and the Performance Index during the iteration.

Having described the high level flow of the adaptation algorithm, we next discuss its details. The inputs at the start of each iteration are:

- Current encoding configuration: (q_0, g_0, r_0, s_0).
- Current encoding attributes: $psnr_0, br_0, fps_0$.
- Performance Index, PI, which is extracted from TPC look up table based on current DTM settings.
- Target boundaries for nominal encoder: $psnr_T, br_T, fps_T$.

Figure 4 shows the flowchart of the adaptation algorithm, which consists of the use of four tasks: *extract, scale, predict,* and *select.* The first three tasks are used to predict the effectiveness of each encoding parameter to satisfy the objective of adaptation, and the last task selects the most effective parameter for the current iteration. We discuss the role of these tasks in the rest of the subsection.

Since we change only one parameter per iteration, we need to study the effect of each parameter on the encoder attributes separately, independent of other parameters. The first task, *extract*, extracts the relationship between each encoding parameter and attribute from the parameters-attribute model, assuming other parameters are constant. Since we have four parameters and three attributes, we get 12 combinations of parameter – attribute extracted functions. For instance $PSNR_Q$, BR_Q, FPS_Q are the extracted functions for quantization parameter. Accordingly, we will have 9 more function for the 3 other parameters. We next show the extraction of $PSNR_Q$ from model data $PSNR(C)$; the same method applies for other parameters and attributes as well.

$$PSNR_Q(q) = PSNR(q, g, r, s)|_{g=g_0, r=r_0, s=s_0}$$

As we mentioned in section III.A, we use experimental model data as a guide for relation between parameters and attributes. However, there may be a difference between the value of an attribute for the current video being encoded and the attribute value according to the extracted function, depending on how different the current video is from the videos that were used for modeling. In the next step (*scale* task), we scale the extracted function by the above mentioned difference. Considering the same example, the scaled function of visual quality versus quantization level, $\widehat{PSNR}_Q$, is calculated as follows:

$$\widehat{PSNR}_Q(q) = \frac{psnr_0}{PSNR_Q(q_0)} \cdot PSNR_Q(q) \quad (4)$$

Similarly, the *scale* task finds the scaled functions corresponding to each of the other 11 extracted functions.

We can now use the scaled functions derived in equation (4) to predict the behavior of encoder parameters and attributes. When all the three scaled functions for one parameter are derived, the third task in Figure 4, *predict*, provides the next value of each parameter. Without loss of generality, we describe the functionality of this task for the quantization parameter, but it applies to all the other parameters as well. In this task, we first calculate three values for quantization by solving the following 3 equations:

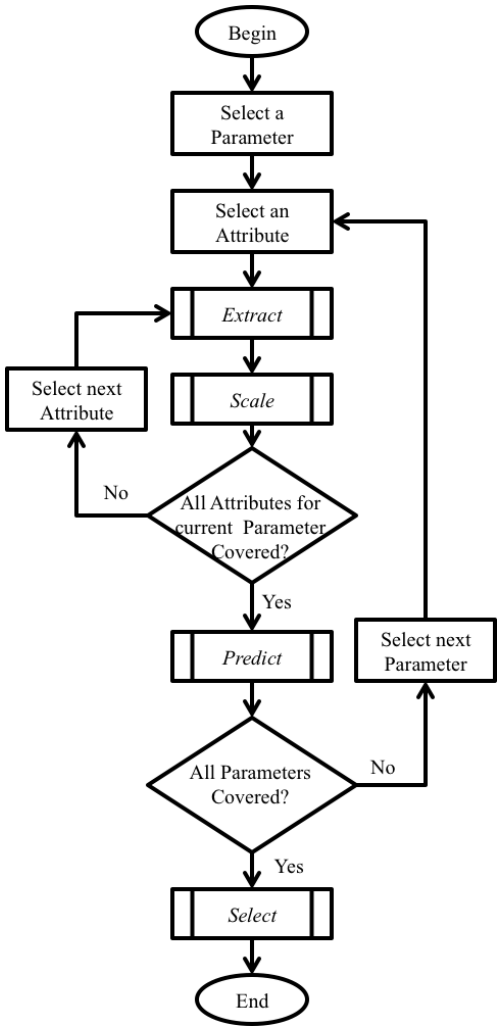

Figure 4. Details of each iteration of the adaptation algorithm

$$\widehat{PSNR}_Q(q) = psnr_T \quad (5.1)$$

$$\widehat{BR}_Q(q) = br_T \quad (5.2)$$

$$PI.\widehat{FPS}_Q(q) = fps_T \quad (5.3)$$

In equation (5.3), the left hand side is multiplied by performance index to reflect the speed impact of the system in presence of DTM. Solving the above equations results in 3 values for quantization, q_1, q_2, q_3. Out of these values, we select one, q_i, that satisfies the following criteria:

$$PI.\widehat{FPS}_Q(q_i) \le fps_T$$
$$q_i \text{ maximizes } \widehat{PSNR}_Q(q) \quad (6)$$
$$q_i \text{ minimizes } \widehat{BR}_Q(q)$$

As shown in Figure 4, the tasks of *extract*, *scale* and *predict* are repeated for all four encoding parameters. After a value of each of the parameters is predicted based on equations (5), they will be compared with each other by the *select* task. The parameter, which maximizes PSNR and

978-1-4673-2154-9/12/$31.00 ©2012 IEEE

TABLE I. IMPLEMENTATION PLATFORM SPECIFICATIONS

Operating System	Mac OS X 1.5.8
Processor	Intel® Core™ 2 Duo
Clock	1.8 GHz
Memory	2 GB

TABLE II. THERMAL MANAGEMENT PROFILES

Profile	**Frequencies (MHz)**	**Throttling Level**
0	1800	N/A
1	1800, 1400	Low
2	1800,1400	High
3	1800, 900	Low
4	1800, 900	High
5	1800, 1600, 1400, 1200, 800	Low
6	1800, 1600, 1400, 1200, 800	High

TABLE III. VOLTAGE VS. FREQUENCY SELECTION

Frequency (MHz)	1800	1600	1400	1200	≤ 900
Voltage (V)	1.175	1.1125	1.05	0.975	0.9

TABLE IV. PERFORMANCE INDEX FOR TEST PLATFORM

Processor Frequency (MHz)	**Performance Index**
1800	100
1600	89.34
1400	77.72
1200	66.70
900	49.01
800	43.56

minimizes bit rate will be selected as the most effective one, thus satisfying the objective of adaptation (equations (1))

The encoder will adapt based on the new parameter, and the attributes of the video stream, and PI corresponding to the new DTM setting will be determined in time for next iteration of the adaptation algorithm. We show the effectiveness of the adaptation algorithm in next section for a DVFS based DTM on Intel® Core™ 2 Duo MacBook laptop.

IV. EXPERIMENTAL RESULTS

In this section, we discuss experimental results using a MacBook Air, with specifications in Table I. Dynamic Thermal Management is done by Cool Book [12], a commercial DVFS tool that allows programming and use of different DVFS profiles for thermal management, consisting of different frequency, voltage, throttling and trigger temperature settings. Trigger temperature is a thermal limit above which the DVFS method will be employed [20].

We have used seven profiles to study the adverse effects of DTM on the video encoder, and the effectiveness of the dynamic adaptation algorithm proposed in this paper to reduce these effects. We set the trigger temperature to 75℃ for all DTM profiles. Profile 0, which is the reference implementation, has no frequency or voltage scaling and processor runs at 1.8GHz (the nominal frequency). The thermal management for this profile is only based on system fan and heat sink. In the rest of the profiles, DVFS is enabled to help thermal management. Table II shows the throttling level and frequency settings used in each thermal management profile, and Table III reports the voltage setting for each frequency. Note the used voltage settings are among the ones predefined and available in the system.

The numbering of the profiles has been selected in ascending order of frequency scaling options. Based on [24], two frequency/voltage levels are sufficient for effective DTM; therefore, profiles 1 to 4 have been selected with only two frequency/voltage levels. Profiles 5 and 6, which use five different frequency/voltage levels, have been selected based on Cool Book recommendations.

In the next subsections, we first present the results of thermal policy characterization (section III.B) of the test platform. Next, we study the performance of the original video encoder, and the proposed dynamically adaptive encoder, when one of the DTM profiles is applied. Finally we present results for the overall performance of the adaptive encoder for different videos and different temperature management profiles.

A. Thermal Policy Management Results

As shown in Table II, our test platform uses six different frequency settings. For each frequency setting, we have encoded several VGA video clips, and measured the performance index according to section III.B. Table IV shows the performance index for each frequency setting.

We observe that the performance indexes for the different settings of the test platform are very close to the ratio of CPU frequency to the nominal hardware frequency. This is because the performance of the x264 encoder implementation is dominated by computation time, and not memory access time, as the latter has been implemented very efficiently.

B. Studying Effects of a DTM Policy and Adaptation

In this subsection, we study in details the effects of one of the DTM policies, and the effectiveness of our proposed dynamic adaptation approach to address DTM effects. We first study the behavior of the original encoder at nominal frequency in the absence of DTM. Then we discuss the effects of DTM on the original encoder performance; and finally we show how the adaptive encoder can significantly improve speed and quality while DTM is effectively managing the temperature. We have selected profile 6, which is the recommended profile for thermal management by Cool Book, and tested it with a video clip from Avatar movie.

Encoding Avatar clip in real time with original x264 video encoder in the absence of DVFS will result in a video stream with PSNR of 41.9 dB and bit rate of 1505 kbps (Figure 5(b)). However, even with processor fan spinning with its maximum speed (6200 rpm), CPU temperature increases up to 91 degree Celsius (Figure 5(a)). Our tests for different video sequences showed that processor temperature could rise up to 94℃ just by encoding a 24 seconds video clip. This shows the need for dynamic thermal management to control the temperature. As shown by Figures 5(c) and (d), in the presence of DVFS (Profile 6) with trigger temperature of 75℃, the processor temperature can be contained to near 75℃ (does not increase over 78℃), but with adverse effects of drop in processor frequency (Figure 5(c)), and a very

978-1-4673-2154-9/12/$31.00 ©2012 IEEE

significant drop in the quality of the encoded video by about 10 dB (Figure 5(d)).

In contrast, Figures 5(e) and (f) show what happens when the proposed adaptive encoder is used while DTM is applied. Figure 5(e) shows that not only the temperature maintained at the desired level of 75℃, but the frequency degradation caused by DVFS is reduced compared to the original encoder without dynamic adaptation (Figure 5 (c)). The above is possible because the adaptation algorithm dynamically changes the computational need of the encoder in response to the real-time changes in the Performance Index affected by DVFS, allowing the processor to cool down faster, and allowing DVFS to use higher frequency levels. Moreover, the adaptation algorithm not only adjusts the encoder according to the changing performance index so as to satisfy real-time speed constraints, but also maximizes video quality with minimum increase in bit rate. This effect is shown in Figure 5(f): as DTM scales down frequency, PSNR drops and bit rate rises but the adaptation algorithm finally balances them and at the end we have a video with PSNR of 39.2, which is very close to what the original encoder achieves.

In summary, without DVFS we can produce high quality video stream but processor temperature increases rapidly. On the other hand, having DVFS in the system helps controlling temperature but it also drops video quality more than 10 dB when frequency scales down. Having adaptive encoder and DVFS together, the video quality does not drop more than 2 dB while processor temperature remains in the safe region. This has come with the price of increased bandwidth about 20%.

C. *Overall Adaptive Encoder Performance*

In this subsection we discuss the performance of adaptation algorithm for three VGA size video clips Avatar, Soccer, and Wildlife. For each video, we first encode them using original video encoder without DVFS (Profile 0) to get the highest video quality to make sure that all the processing capacity is used by the encoder. Then we used the PSNR and bit rate of those videos as the reference when encoding that specific clip with original or adaptive video encoder in the presence of dynamic thermal management.

Clearly, adding adaptation algorithm to the original encoder introduces some overhead to the encoding process. It takes some time to calculate the parameters and reconfigure the encoder and this time is taken from the total time budget available for the encoding process. Adaptation algorithm considers this time difference and changes the encoder parameters to compensate it. Therefore the time (or the computation capacity) taken for adaptation algorithm will be translated into PSNR drop and/or bit rate increase of encoded stream. On average, the overhead due to adding adaptation algorithm to the original encoder is equivalent to 0.26 dB drop in PSNR and 8.3% increase in bit rate.

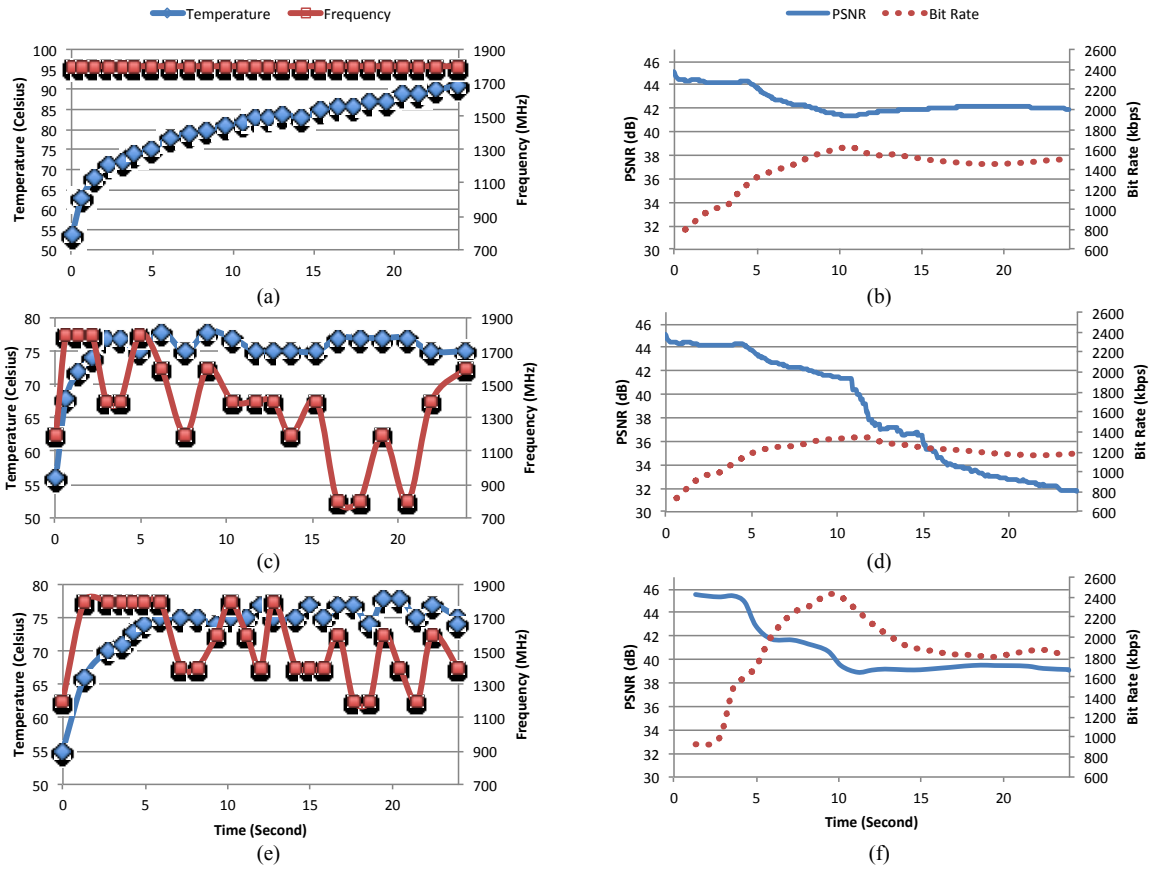

Figure 5. (a) Temperature and frequency of original encoder for Profile 0, (b) PSNR and bit rate of original encoder for Profile 0, (c) Temperature and frequency of original encoder for Profile 6, (b) PSNR and bit rate of original encoder for Profile 6, (e) Temperature and frequency of adaptive encoder for Profile 6, (f) PSNR and bit rate of adaptive encoder for Profile 6, while encoding Avatar video clip in real-time.

978-1-4673-2154-9/12/$31.00 ©2012 IEEE

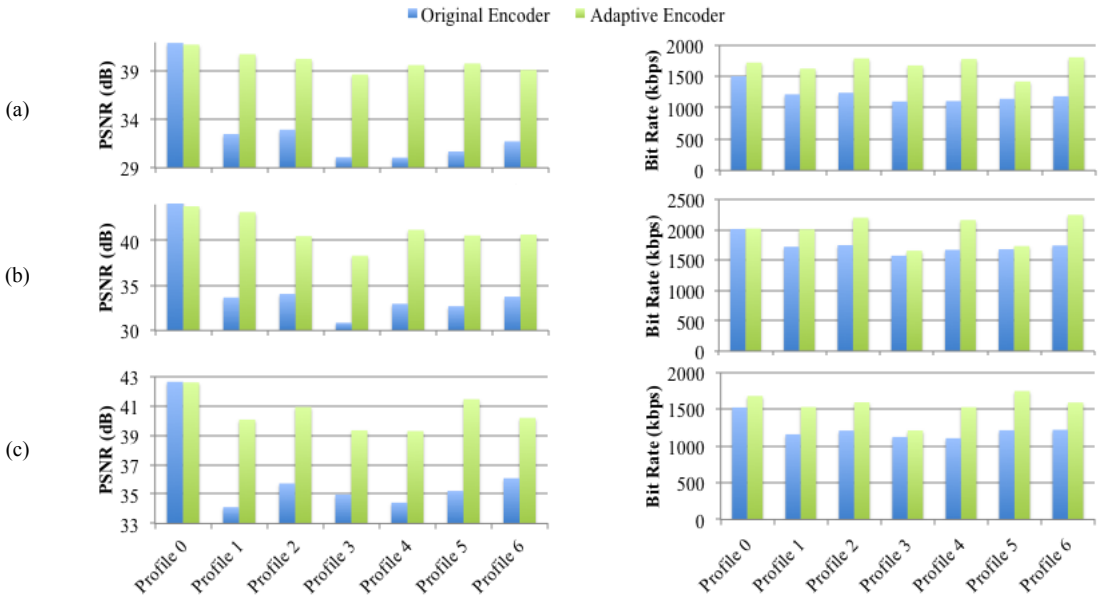

Figure 6. PSNR and bit rate of (a) Avatar video clip, (b) Soccer video clip, and (c) Wild Life video clip, when original and adaptive video encoder is used for real-time encoding, using DTM Profiles 0-6.

As shown in Figure 6, the proposed adaptation algorithm has been able to improve video quality compared to original encoder for all the DVFS profiles. While video quality of the original encoder drops about 10dB on average, video quality of adaptive encoder has dropped by an average of only 2.4 dB, with marginal bitrate increase of about 4%.

Finally we observe from Figure 6 that the best DVFS policy for adaptive encoder is policy 2 (2 frequency/voltage levels with 12% frequency derate), where average video quality drop has been 1.6 dB with marginal 2.7% bit rate increase.

V. CONCLUSION

Due to limitations of cooling and increase in power density, use of Dynamic Thermal Management in electronic systems is inevitable; however, DTM impacts the performance of systems, which can be particularly problematic for compute intensive and real-time applications such as video encoding, which are the same applications that may need DTM the most. In this paper, we focused on H.264 video encoding as the application, and showed that DTM can have an unacceptably high impact on the performance and quality of video encoding. We presented an approach which can dynamically adapt the encoder tasks in response to the DTM performance impact, such that the encoder can perform in real time, and with significantly less effect on the encoding quality, all at marginal increases in encoding bit rate. We demonstrated the approach using a commercial DVFS based DTM tool on an Intel® Core™2 Duo processor. We will also investigate the applicability of the propsoed approach on other compute intensive and real-time applications like graphics rendering, and other platforms like mobile and multi-core server platforms.

ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation under Grant No. CNS-0917354.

REFERENCES

[1] D. Shin; S. W. Chung; E.Y. Chung; N. Chang; "Energy-Optimal Dynamic Thermal Management: Computation and Cooling Power Co-Optimization," *Industrial Informatics, IEEE Transactions on* , vol.6, no.3, pp.340-351, Aug. 2010.

[2] A. Kumar; Li Shang; Li-Shiuan Peh; Jha, N.K.; , "System-Level Dynamic Thermal Management for High-Performance Microprocessors," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* , vol.27, no.1, pp.96-108, Jan. 2008

[3] Yang Ge; Qinru Qiu; , "Dynamic thermal management for multimedia applications using machine learning," *Design Automation Conference (DAC), 2011 48th ACM/EDAC/IEEE* , vol., no., pp.95-100, 5-9 June 2011.

[4] Pakbaznia, E.; Ghasemazar, M.; Pedram, M.; , "Temperature-aware dynamic resource provisioning in a power-optimized datacenter," *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2010* , vol., no., pp.124-129, 8-12 March 2010

[5] Wei Huang; Allen-Ware, M.; Carter, J.B.; Elnozahy, E.; Hamann, H.; Keller, T.; Lefurgy, C.; Jian Li; Rajamani, K.; Rubio, J.; , "TAPO: Thermal-aware power optimization techniques for servers and data centers," *Green Computing Conference and Workshops (IGCC), 2011 International* , vol., no., pp.1-8, 25-28 July 2011.

[6] Kevin Skadron, Mircea R. Stan, Karthik Sankaranarayanan, Wei Huang, Sivakumar Velusamy, and David Tarjan,"Temperature-aware microarchitecture: Modeling and implementation". *ACM Trans. Archit. Code Optim.* 1, 1 (March 2004), 94-125.

[7] Ayse Kivilcim Coskun, "Efficient Thermal Management for Multiprocessor Systems" , *University of California, San Diego*, 2009.

[8] Srinivasan, V.; Hermerding, J.G.; Khanna, R.; , "An innovative approach to dynamic platform and thermal management for Mobile

platforms," *Energy Aware Computing (ICEAC), 2010 International Conference on* , vol., no., pp.1-4, 16-18 Dec. 2010

[9] Sushu Zhang; Chatha, K.S.; , "Thermal aware task sequencing on embedded processors," *Design Automation Conference (DAC), 2010 47th ACM/IEEE* , vol., no., pp.585-590, 13-18 June 2010

[10] Cisco, "Cisco Visual Networking Index: Forecast and Methology, 2010-2015", June 2011.

[11] x264. *VideoLan.* [Online] [Cited: January 27, 2012.] http://www.videolan.org/developers/x264.html.

[12] Cool Book by Mangus Lundholm. [Online] [Cited: January 27, 2012.] http://coolbook.se/CoolBook.html.

[13] Hang Liu and Magda El Zarki. "Adaptive source rate control for real-time wireless video transmission". *Mob. Netw. Appl.* 3, 1 (June 1998), 49-60.

[14] Osama Lotfallah, Martin Reisslein, and Sethuraman Panchanathan. "Adaptive bitstream switching of pre-encoded PFGS video." In *Proceedings of the ACM workshop on Advances in peer-to-peer multimedia streaming* (P2PMMS'05). ACM, New York, NY, USA, 11-20.

[15] Y.V. Ivanov, C.J. Bleakley, "DynamicComplexity Scaling for Real-Time H.264/AVC Video Encoding*,"15th international conference on Multimedia.* pp. 962 – 970, 2007.

[16] Y.V. Ivanov, C.J. Bleakley, "Real-Time H.264 Video Encoding in Software with Fast Mode Decision and Dynamic Complexity Control"*,ACM Transaction on Multimedia Computing, Communication & Application: TOMCCAP*, pp. 5-5:21, 2010.

[17] M. Shafique, L. Bauer, J. Henkel, "enBudget: A Run-Time Adaptive Energy-Budgeting Scheme for Energy-Aware Motion Estimation in H.264/MPEG-4 AVC Video Encoder*",Design Automation & Test in Europe.* pp. 1725-1730, 2010.

[18] Inchoon Yeo; Eun Jung Kim; , "Hybrid dynamic thermal management based on statistical characteristics of multimedia applications," *Low Power Electronics and Design (ISLPED), 2008 ACM/IEEE International Symposium on* , vol., no., pp.321-326, 11-13 Aug. 2008.

[19] Inchoon Yeo; Heung Ki Lee; Eun Jung Kim; Ki Hwan Yum; , "Effective Dynamic Thermal Management for MPEG-4 decoding," *Computer Design, 2007. ICCD 2007. 25th International Conference on* , vol., no., pp.623-628, 7-10 Oct. 2007

[20] Wonbok Lee; Patel, K.; Pedram, M.; , "GOP-Level Dynamic Thermal Management in MPEG-2 Decoding," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on* , vol.16, no.6, pp.662-672, June 2008

[21] Hanumaiah, V.; Vrudhula, S.; , "Temperature-aware DVFS for Hard Real-time Applications on Multi-core Processors," *Computers, IEEE Transactions on* , vol.PP, no.99, pp.1, 0

[22] Forte, Domenic; Srivastava, Ankur; , "Energy and thermal-aware video coding via encoder/decoder workload balancing," *Low-Power Electronics and Design (ISLPED), 2010 ACM/IEEE*

[23] Wonbok Lee; Kimish Patel; Massoud Pedram; , "Dynamic Thermal Management for MPEG-2 Decoding," *Low Power Electronics and Design, 2006. ISLPED'06. Proceedings of the 2006 International Symposium on* , vol., no., pp.316-321, 4-6 Oct. 2006.

[24] Skadron, K.; , "Hybrid architectural dynamic thermal management," *Design, Automation and Test in Europe Conference and Exhibition, 2004. Proceedings* , vol.1, no., pp. 10- 15 Vol.1, 16-20 Feb. 2004.

978-1-4673-2154-9/12/$31.00 ©2012 IEEE