Joint Work and Voltage/Frequency Scaling for Quality-Optimized Dynamic Thermal Management

Ali Mirtar, *Member, IEEE*, Sujit Dey, *Fellow, IEEE*, and Anand Raghunathan, *Fellow, IEEE*

***Abstract*— Dynamic thermal management (DTM) is commonly used to ensure reliable and safe operation in modern computing systems. DTM techniques are based on slowing down or shutting down parts of a system; hence, they effectively reduce system performance and thereby adversely impact applications. In this paper, we focus on real-time applications in which degradation in performance translates to a loss in application quality, and address the problem of quality-optimized DTM, wherein the objective of DTM is to satisfy specified temperature constraints while optimizing application quality metrics. We first introduce a new DTM method called dynamic work scaling (DWS), which is based on modulating an application's computational requirements. Next, we observe that application quality and platform temperature are effectively determined by two key parameters, viz., the application's computational requirement and the platform's computing capacity, and formulate the relationship between them. Finally, we propose a quality-optimized DTM based on joint dynamic work and voltage/frequency scaling (DWVFS). We have implemented the proposed DTM technique and evaluated it for two applications: 1) H.264 video encoding and 2) turbo decoding. Our results demonstrate that DWVFS can provide superior results in terms of application quality compared with both DVFS and DWS-based DTM at identical temperature constraints.**

***Index Terms*— Application adaptation, dynamic thermal management, dynamic voltage scaling.**

I. Introduction

THE increase in complexity of integrated systems, together with an inability to sustain classical scaling have substantially elevated power density and temperature concerns in computing platforms [1]–[21]. Operating under high temperature reduces the reliability and durability of chips [3]. To avoid these adverse effects, cooling systems, such as heat sinks and fans, have been used since early generations of computing platforms. However, either due to battery or size limitations for mobile devices or due to very high cooling requirements for servers and datacenters, conventional cooling methods alone cannot fully address the thermal needs of modern designs. Therefore, dynamic thermal management (DTM) is used either as a supplement or as a replacement for cooling methods.

Many DTM methods have been developed for general purpose processors [6], [7], servers and data centers [4], [5], mobile platforms [8], and embedded systems [9]. Even though they vary in their approaches, a common attribute is that they all negatively affect the performance of the computing platform, leading to an increase in the runtime of tasks. In the case of many real-time applications, longer runtimes result in a loss of the application's functional quality, which we broadly define as a metric of how well an application is performing the task that it is supposed to perform. For instance, we define the functional quality of a video encoder as the visual quality of the video; the functional quality of a turbo decoder is defined as its effective decoding throughput. The efficacy of previous DTM methods has been mostly quantified by the runtime increase [1]–[21].

In this paper, we directly focus on optimizing the functional quality of a real-time application rather than its tasks' runtimes. We observe that complex real-time applications, such as video encoding, gaming (3-D rendering), communication coding, and so on, present parameters that can be used to tune their computing requirement, i.e., the workload presented to the computing platform. Based on this insight, we first introduce a DTM method called dynamic work scaling (DWS), an application level DTM technique in which we scale the computing requirements of an application by tuning its parameters, and hence the workload that it presents, to manage the temperature of the underlying platform. Then, we motivate and propose a hybrid method based on DWS and conventional dynamic voltage and frequency scaling (DVFS). The proposed approach is called joint dynamic work and voltage/frequency scaling (DWVFS). We formulate this hybrid DTM as a multidimensional constrained optimization problem and present an analytical solution. Furthermore, we propose a fast algorithm to solve the optimization problem in real time to perform quality-optimized DTM. Overall, the contributions of this paper are twofold:

1) viewing DTM from the lens of its impact on an application's functional quality, and formulation of quality-optimized DTM as a constrained optimization problem;
2) proposal of a hybrid DTM technique based on joint DWS and DVFS, and demonstration of its benefits over conventional DTM techniques.

In the rest of this paper, we first review related work on dynamic thermal management (Section II). Then, we introduce DWS as a DTM method and show its utility in thermal

Manuscript received December 21, 2013; revised April 15, 2014; accepted June 9, 2014. Date of publication August 28, 2014; date of current version May 20, 2015. This work was supported by the National Science Foundation under Grant CNS-0917354.

A. Mirtar and S. Dey are with the Department of Electrical and Computer Engineering, University of California at San Diego, La Jolla, CA 92093 USA (e-mail: amirtar@ucsd.edu; sdey@ucsd.edu).

A. Raghunathan is with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907 USA (e-mail: raghunathan@purdue.edu).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors. This includes a document file, which contains the proofs of theorems mentioned in the manuscript. This material is 251 KB in size.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2014.2333741

management (Section III). We then formulate quality-optimized DTM as a constrained optimization problem, discuss the relationship between work and voltage/frequency scaling with the application's quality and the platform's temperature and describe an analytical solution (Section IV). Next, we present a fast algorithm to perform quality-optimized DTM using DWVFS in real-time (Section V). Finally, we apply the proposed DTM approach to two real-time applications—a video encoder and a turbo decoder—and compare its efficacy with conventional DTM methods (Section VI).

II. Related Work

A variety of techniques have been proposed for DTM, including instruction fetch gating [10], DVFS [10], [13], scheduling [11], [12], [14], [34]–[38], and thermal aware load balancing [14]. Many hybrid methods also have been introduced to combine the benefits of multiple approaches and to avoid their drawbacks [10]–[15].

Application-specific DTM methods have been extensively studied in the context of video encoding/decoding applications [3], [16]–[21]. In these studies, common DTM techniques were adopted and the timing characteristics of video encoding/decoding applications were used to guarantee real-time execution.

The main difference between this paper and most of the above efforts [1]–[21], [34]–[38] arises from the objective function. In previous work, the objective has been to maximize performance in the presence of thermal constraints. Note that the term performance has been quantified by application run time, response time, whether deadlines are met, or other timing metrics of an application. Instead, we propose to look at the end user experience and maximize the functional quality that is required from an application. This matters especially in the context of real-time applications where different timing violations (e.g., dropping of different frames in a video stream) may affect the application's quality differently. In addition, we use an application layer approach compared with scheduling approaches which are OS level methods [11], [12], [34]–[38]. This enables us to exploit application properties to design a quality-aware method, compared with other methods [11], [12], [15], [34]–[38] which are oblivious to their effects on application quality and only optimize for timing metrics. Furthermore, we change the application itself to not only help with thermal management, but also improve application quality while computing capacity is limited. In contrast, other scheduler-based methods affect the order and the time in which tasks are being executed and they do not modify the application tasks themselves. Another contribution of this paper is a more generic formulation and treatment of DWS for real-time applications. Previous DTM methods [3], [16]–[21] that have been applied to just real-time video encoding/decoding are dependent on application specifics. Therefore, they cannot easily be adapted to other real-time applications. In this paper, we formulate the relationships between application-level functional quality and temperature on one hand and the computational requirement and computing capacity on the other, and utilize them to propose a DTM technique that is applicable to a broad range of real-time applications.

III. DWS: A Knob for Thermal Management

As we mentioned, many hardware and software (scheduling) methods have been developed and used for DTM. These methods may adversely affect the quality of the application running on the system[1] by reducing the system's performance; however, they do not actually change the running application itself. In this section, we introduce a new method of thermal management by modifying an application's tasks through its parameters. This modification results in changing the computing requirements of a real-time application running on the system, and thereby changing the application's workload on the system.

There are many real-time and complex applications whose computing requirements can be controlled by their parameters. For example, quantization level for H.264 video encoder [22], view distance for 3-D graphics rendering [23], and number of decoding iterations for a turbo decoder [24] in a Software Defined Radio [25] can affect the computing requirements of the application. We study the effects of workload change on a system's temperature in Section III-A and then we briefly discuss how changing these parameters can affect user experience and the application's functional quality.

A. DWS Concepts

In real-time applications, specific tasks should be completed in a given time interval or within a deadline. For instance, in video encoding with the frame rate of 30 frames/s, all the tasks required for encoding a frame should be completed in less than a $1/30$th of a second. In such applications, if the platform does not have enough computing capacity for the application's tasks to finish on time, the functional quality of the application would be degraded. On the other hand, if the system's computing capacity is more than the computing requirements of the application, the platform is not fully utilized. Modern computing platforms have various hardware techniques that exploit even fine-grained intervals of less-than-complete utilization for power reduction (e.g., through clock gating). Note that this is orthogonal to DVFS techniques, which are applied at a coarser granularity and under software control. This reduced power dissipation under lower utilization in turn leads to lower thermal load. This characteristic of real-time applications is the foundation of the DWS. We try to control the computing requirements of a real-time application and thereby tradeoff the functional quality of the application against the thermal load that it generates.

Thermal dynamics of junction temperature in an integrated circuit can be described as follows [26]:

$$\Theta' = \frac{P}{C_{\text{th}}} - \frac{\Theta}{C_{\text{th}} \cdot R_{\text{th}}} \tag{1}$$

where P is the power consumed in the silicon, C_{th} is the thermal capacitance of the silicon, R_{th} is the thermal resistance

[1]The two terms platform and system have been used interchangeably in this paper.

DVFS:	DWS:
Every 100 millisecond:	Every 100 millisecond:
Θ= Platform temperature	Θ= Platform temperature
If $\Theta > \Theta_c$: Decrease frequency and voltage one step	If $\Theta > \Theta_c$: Decrease computing requirements one step
Else: Increase frequency and voltage one step	Else : Increase computing requirements one step

Fig. 1. Greedy implementation of DVFS and DWS.

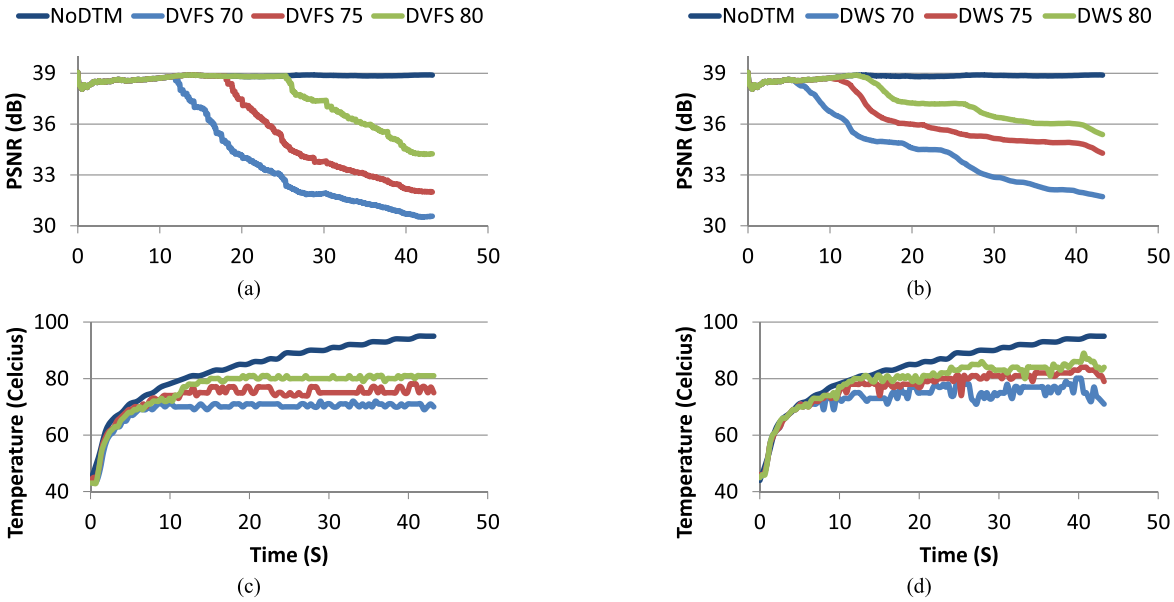

Fig. 2. Effects of DWS and DVFS on an H.264 video encoder application's functional quality and platform's temperature. (a) Visual quality for DVFS. (b) Visual quality for DWS. (c) Platform temperature with DVFS. (d) Platform temperature with DWS.

of the silicon, Θ is the silicon junction temperature relative to the ambient temperature, and Θ' is the rate of change in junction temperature.

In a short time window that power is constant, the solution of (1) is

$$\Theta(t) = R_{\text{th}}P + (\Theta_i - R_{\text{th}}P)e^{-\frac{t}{\tau}} \tag{2}$$

where Θ_i is the initial temperature and $\tau = R_{\text{th}}C_{\text{th}}$ is the thermal time constant.

We formulate the periodic characteristic of real-time applications mentioned earlier as follows. The time interval in which the application needs to perform a set task is called T. For instance, T can be the required time to encode a frame in case of real time video encoding, or the time needed to decode a packet of data in the case of turbo decoder in a wireless communication. We define the stress ratio, u, as the percentage of the time that the application is stressing the platform, where $0 \leq u \leq 1$. Therefore, the percentage of the time that system is in rest is $(1-u)$. Hence, u is equal to one when the computing requirement of the application is equal or more than the computing capacity of the platform.

We define the average power consumed during the application stress time as P_s and the average power consumed during the rest time as P_r (where $P_r < P_s$). In practice, the time interval T for real-time applications is in the order of microseconds or milliseconds; however the thermal time constant is usually in the order of seconds ($T \ll \tau$). With this assumption, it can be proven that the platform would reach to a steady temperature θ_{ss} that is in linear relationship with application stress ratio.

$$\Theta_{\text{ss}}(u) = R_{\text{th}}(P_r(1-u) + P_s u). \tag{3}$$

The proof of equation (3) is shown in an appendix provided in supplementary material.

In summary, the parameters of a real-time application can be used to change its computing requirements and hence its work load. Thereby, the portion of time that the application stresses the platform can be affected. Furthermore, the steady temperature of the platform is a linear function of the stress ratio. This means by controlling the application's workload through changing its parameters, one can control and manage the temperature of the platform (DTM).

B. DWS Implementation

As we mentioned, work scaling is a potential control knob for thermal management. In this section, we define the objective of DWS as a DTM technique, and by using a greedy implementation will show its effectiveness.

Since changing the work load of a real-time application by scaling its computational requirements/complexity can also affect its functional quality, we define the objective of DWS with respect to application function quality as follows.

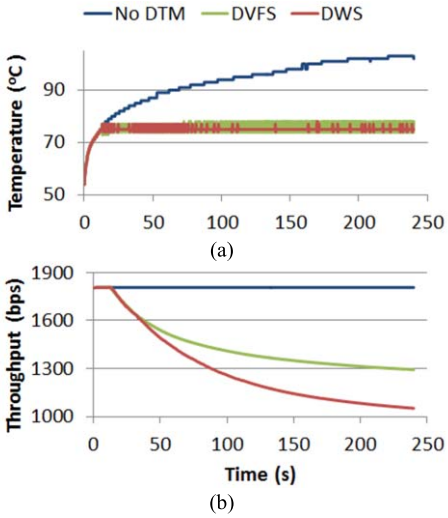

Fig. 3. Using DVFS and DWS for turbo decoder application. (a) Platform temperature in degree Celsius. (b) Turbo decoder throughput in bps.

The objective of DWS for thermal management is to maximize the functional quality of the real-time application while making sure the temperature of the platform does not exceed a given critical temperature.

The above statement can be described in the following mathematical form:

$$\text{Maximize: } Q(p) \quad \text{s.t.: } \Theta(p) \leq \Theta_c$$

where Q is the application's functional quality, and Θ is the platform's temperature, as a function of the application's parameter p. Therefore, the control variable in the DWS is the application parameter that would vary application computing requirement and workload.

Fig. 1 shows a greedy implementation of DWS and DVFS for DTM. Both DWS and DVFS algorithms will control platform temperature (perform DTM), but they use different control variables. We have used the DTM algorithms on two applications, H.264 video encoder and turbo decoder. Both DTM schemes were evaluated for these two applications executing on a Macbook Alr platform with the Intel Core2Duo processor. In the case of the H.264 video encoder, the application parameter p is constant rate factor (CRF), and the functional quality Q is the encoded video stream's peak Signal-to-noise ratio (PSNR). The number of iterations in the decoding loop is the application parameter for turbo decoder and its effective throughput is its functional quality. The details and description of these applications, their parameters and their functional quality along with the platform specifications are provided in the results section (Section VI).

Fig. 2 compares the results of DWS and DVFS implementations described in Fig. 1 for four test conditions. One test is without any DTM and the other three tests are with 70 °C, 75 °C, and 80 °C as critical temperature. Fig. 2(a) and (b) shows the visual quality of the encoded stream (in PSNR) over time for DVFS and DWS, respectively. It is clear that in this example, quality drop in DVFS is faster than DWS and eventually, the video stream produced under DWS has a better quality (higher PSNR). On the other hand, from Fig. 2(c) and (d) we observe that DVFS controls the temperature better than DWS meaning the violation of critical temperature occurs less frequently.

Fig. 3 shows the result of using DWS and DVFS for a turbo decoder application for a critical temperature of 75 °C. Fig. 3(a) shows that both DWS and DVFS can control the temperature around the target temperature while without DTM, the temperature rises above 100 °C. Fig. 3(b) shows the effects of each DTM method on the functional quality which in this case is decoder's effective throughput. In this scenario, the negative effect of DWS on application quality is more than DVFS.

These examples are provided as proofs of concept to show that DWS can be used as a DTM tool in practice. In addition, in the case of video encoder, DWS produces better quality results than DVFS while in the case of turbo decoder, DVFS produces better functional quality. The results tell us that while both DWS and DVFS can be used for DTM, their effectiveness and impact on application quality can be different under different application conditions. We investigate these conditions in the rest of this paper and study the joint effects of DWS and DVFS on both application's functional quality and platform temperature.

IV. Quality Optimized DTM

Our goal is to provide a method that best utilizes both DWS and DVFS in order to obtain a quality optimized DTM methodology. To achieve this goal, we first study the joint effects of DWS and DVFS on temperature as well as functional quality of a real-time application. Next, we formulate quality-optimized DTM in the form of a constrained optimization problem, whose solution is the new joint DWVFS approach for DTM.

A. Quality and Thermal Contour Lines

We start with some definitions that will help in quantifying different behaviors of any real-time application in general terms, and eventually develop a generic DTM algorithm. Changing a platform's voltage/frequency affects the computing capacity of the platform. Therefore, we define our first variable as the computing capacity or C_c, which represents the effect of DVFS on the platform. On the other hand, when we change a parameter of an application, we affect a change in the amount of computation requested from the platform. Therefore, we define our second variable as the application's computing requirements or C_r to represent the effect of DWS. Hence, the application's functional quality and platform's temperature can be described as functions of these two variables

$$\text{Application functional quality: } Q(C_c, C_r)$$
$$\text{Platform temperature: } \Theta(C_c C_r).$$

Since both $Q(C_c C_r)$ and $\Theta(C_c C_r)$ are functions of two variables, we use contour lines to study their general behavior. A contour line (or isoline) of a function with two variable, is a curve in which the value of the function is the same [27]. Contour lines can be used to show the general behavior of a two-variable function like its extremums or rate of change.

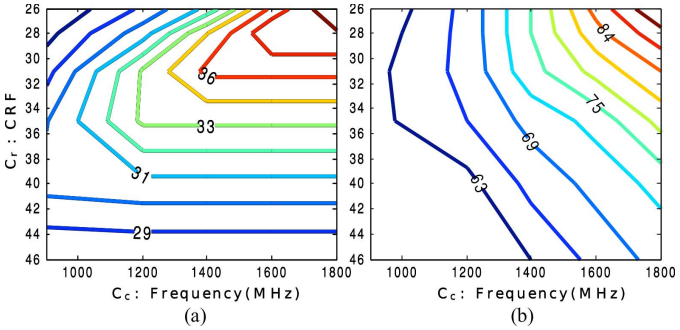

Fig. 4. Contour plots of an H.264 encoder. (a) Functional quality in PSNR. (b) Platform temperature in degree Celsius.

For example, Fig. 4 shows the contour lines plots of $Q(C_cC_r)$ and $\Theta(C_cC_r)$ for an H.264 video encoder application. The horizontal axis is the CPU frequency; as frequency increases, the computing capacity of the platform increases. The vertical axis is the CRF, which is one of the H.264 video encoder's parameters. The increase in CRF decreases the computing requirements of the encoder (more information about CRF is provided in Section VI). These plots illustrate that peak quality and temperature occur at minimum CRF and maximum CPU frequency. We first derive the general shape of temperature contour lines and then study functional quality contour lines. Then, we will use them in the next section to formulate the quality optimized dynamic thermal management.

1) Temperature Function: As we mentioned earlier, platform temperature is a function of C_c and C_r. In addition, it is self-evident that increasing computing requirements of application increases the platform utilization as long as computing capacity is available

$$\begin{cases} \dfrac{\partial u}{\partial C_r} = 0 & C_r \geq C_c \\ \dfrac{\partial u}{\partial C_r} > 0 & C_r < C_c. \end{cases} \tag{4}$$

From (3) (Section III-A) and (4), we derive that by increasing computing requirements while computing capacity is available, the temperature increases

$$\begin{cases} \dfrac{\partial \Theta}{\partial C_r} = 0 & C_r \geq C_c \\ \dfrac{\partial \Theta}{\partial C_r} > 0 & C_r < C_c. \end{cases} \tag{5}$$

The same type of relationship holds between temperature and computing capacity. Computing capacity is proportional to platform's frequency and the platform's temperature increases by increasing its frequency and voltage

$$\frac{\partial \Theta}{\partial C_c} > 0. \tag{6}$$

Equations (5) and (6) provide us with enough information to find out the general thermal behavior of a platform. In fact, the contour lines characteristics can be extracted from the function's partial derivatives as described in the following corollaries.

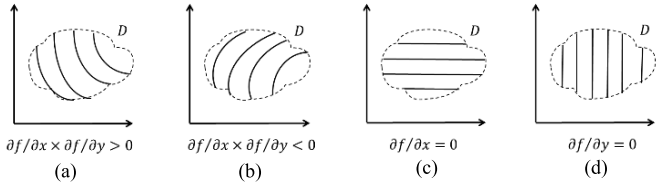

Fig. 5. General shapes of contour lines with respect to the function's partial derivatives with the conditions specified in (a) Corollary 1, (b) Corollary 2, (c) Corollary 3, and (d) Corollary 4.

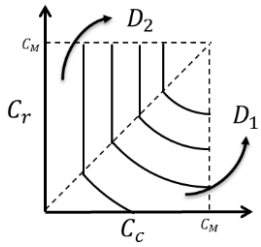

Fig. 6. Contour lines plot of platform's temperature executing a real-time application with respect to computing capacity and computing requirements. The diagonal dashed-line represents $C_r = C_c$.

The contour lines plot of a two-variable function $f(x, y)$ in a domain (D), where the sign of the partial derivatives does not change and at least one of them is nonzero, will be (Fig. 5).

Corollary 1: Noncrossing decreasing lines if $\partial f/\partial x \times \partial f/\partial y > 0$.

Corollary 2: Noncrossing increasing lines if $\partial f/\partial x \times \partial f/\partial y < 0$.

Corollary 3: Parallel horizontal lines if $\partial f/\partial x = 0$.

Corollary 4: Parallel vertical lines if $\partial f/\partial y = 0$.

The proofs of above corollaries are shown in an appendix provided in supplementary material. By applying the above corollary on (5) and (6), we come up with the general thermal behavior of a real-time application. We define the maximum computing capacity of the platform as C_M. Therefore, the domain of temperature function D is defined as

$$D = \{(C_c, C_c) | 0 \leq C_c \leq C_M, 0 \leq C_r \leq C_M\}.$$

This domain does not fall into any of the categories mentioned in the corollary. Therefore, we break it into two sub domains

$$\begin{aligned} D_1 &= \{(C_c, C_c) | 0 \leq C_c \leq C_M, \quad 0 \leq C_r < C_c\} \\ D_2 &= D - D_1. \end{aligned} \tag{7}$$

The first domain, D_1, follows Corollary 1 formulation and the second domain, D_2, follows Corollary 4 formulation (5) and (6). Fig. 6 displays the general shape of the temperature contour lines of a real-time application where the contour lines in D_1 and D_2 are made from Fig. 5, Corollary 1 and 4 plots.

2) Quality Function: We start analysis of the quality function, $Q(C_cC_r)$, with the subdomain defined in (7), D_1. In this subdomain, the computing requirement is smaller than computing capacity of the platform. Therefore, a slight change in the computing capacity will not affect the quality of the application. This phenomenon can be represented

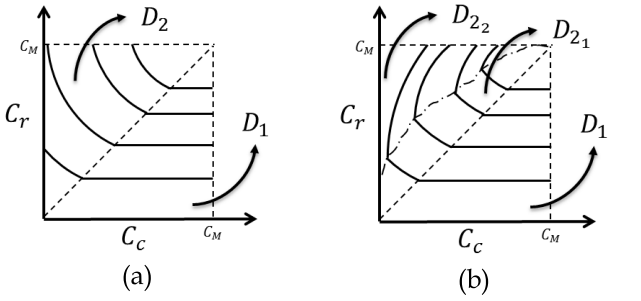

Fig. 7. Contour lines plot of a real-time application's quality function Q with respect to computing capacity and computing requirements. The diagonal dashed line represents $C_r = C_c$. (a) Quality contour lines plot when $\partial Q/\partial C_r > 0$ for all the points in D_2. (b) Quality contour lines plot when $\partial Q/C_r$ changes sign in D_2. The dashed-dot line represents these points $(C_r = f(C_c))$.

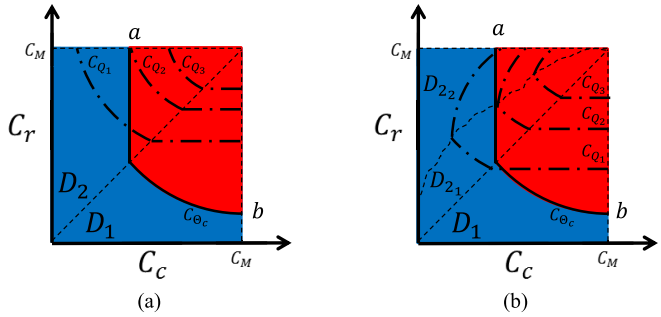

Fig. 8. Superimposition of temperature and quality contour lines plots. Solid line is the contour plot of the platform's temperature at critical temperature. The dashed-dot lines are functional quality contour lines. The points in the red area will violate the thermal constraint and conversely, the points in the blue area will not violate the thermal constraint. (a) When $\partial Q/\partial C_r(C_c, C_r)$ does not change sign. (b) When $\partial Q/\partial C_r(C_c, f(C_c))$ changes sign.

mathematically as

$$\frac{\partial Q}{\partial C_c} = 0, \quad \text{if: } C_r < C_c. \tag{8}$$

Using Corollary 3, we conclude that the contour lines of the quality function are horizontal lines in D_1 (Fig. 7). Next, we study the D_2 domain where the computing capacity is less than computing requirements. Let us consider an arbitrary point in this domain. Since the computing capacity is less than the computing requirements, the functional quality of application would be less than expected. Now, if the computing capacity drops, the functional quality suffers even more due to higher limitations induced by the platform. Therefore, we can derive

$$\frac{\partial Q}{\partial C_c} > 0, \quad \text{if: } C_r > C_c. \tag{9}$$

Now, we take a look at computing requirements of an application and its role on the application's quality. An application is usually designed in a way that the increase in computing requirements is going to increase the application's functional quality unless it faces any limitations from the platform. It means

$$\frac{\partial Q}{\partial C_r} > 0, \quad \text{while: } C_r < C_c \tag{10}$$

when $C_r > C_c$, the effects of C_r change on quality function would be complex. One would expect to see higher quality in response to C_r increase; however, due to limited computing capacity, not only we may not achieve as much quality increase as we expected, but we may also lose quality due to application tasks that may not be completed. We know the rate of quality change with respect to computing requirement is positive right before entering D_2 (10). Therefore, $\partial Q/\partial C_r$ in D_2 is either always positive, or it starts positive and changes to negative at some point in D_2. We show the points where the derivative sign changes with the following notation:

$$C_r = f(C_c) \quad \text{for } (C_c, C_r) \in D_2. \tag{11}$$

In summary, there would be two scenarios. First scenario is when $\partial Q/\partial C_r > 0$ for all the points in D_2 (13.1). Using (9) and Corollary 1, we conclude that the general shape of quality contour lines would be in the form of Fig. 7(a) for this scenario. The second scenario would be if the sign of $\partial Q/\partial C_r$ changes from positive to negative at some points in D_2 described by (11). Therefore, we divide D_2 into two smaller subdomains based on (11)

$$\begin{aligned} D_{2_1} &= \{(C_c, C_r) | 0 \le C_c \le C_M, \quad C_c < C_r < f(C_c)\} \\ D_{2_2} &= D_2 - D_{2_1}. \end{aligned} \tag{12}$$

Accordingly, the application quality change with respect to computing capacity will be (13.2)

$$\frac{\partial Q}{\partial C_r} > 0 \quad \text{while } C_c < C_r < C_M \tag{13.1}$$

$$\begin{cases} \dfrac{\partial Q}{\partial C_r} > 0 & \text{while } C_c < C_r < f(C_c) \\ \dfrac{\partial Q}{\partial C_r} < 0 & \text{while } f(C_c) < C_r < C_M. \end{cases} \tag{13.2}$$

Using (9), Corollary 1 and Corollary 2, we conclude that the general shape of quality contour lines would be in the form of Fig. 7(b) for the second scenario. Next, we formulate quality optimized DTM into an optimization problem and solve it using the contour plots we introduced in this section.

B. Quality and Thermal Management: Problem Formulation

As we mentioned earlier, we are interested to come up with a quality optimized DTM based on joint DWVFS. The objective of DWVFS for DTM can be described in the following statement.

The objective of DWVFS for thermal management is to maximize the functional quality of the real-time application while making sure the temperature of the platform does not exceed a given critical temperature.

The above statement can be described in the following mathematical form:

$$\text{Maximize: } Q(C_c, C_r) \quad \text{s.t: } \Theta(C_c, C_r) \le \Theta_c. \tag{14}$$

In the rest of this section, we derive a solution for the above optimization problem. The proofs of lemmas and theorems are shown in an appendix provided in supplementary material.

Lemma 1: The solution of the optimization problem is located on the critical temperature contour line:

$$\Theta(C_c, C_r) = \Theta_c. \tag{15}$$

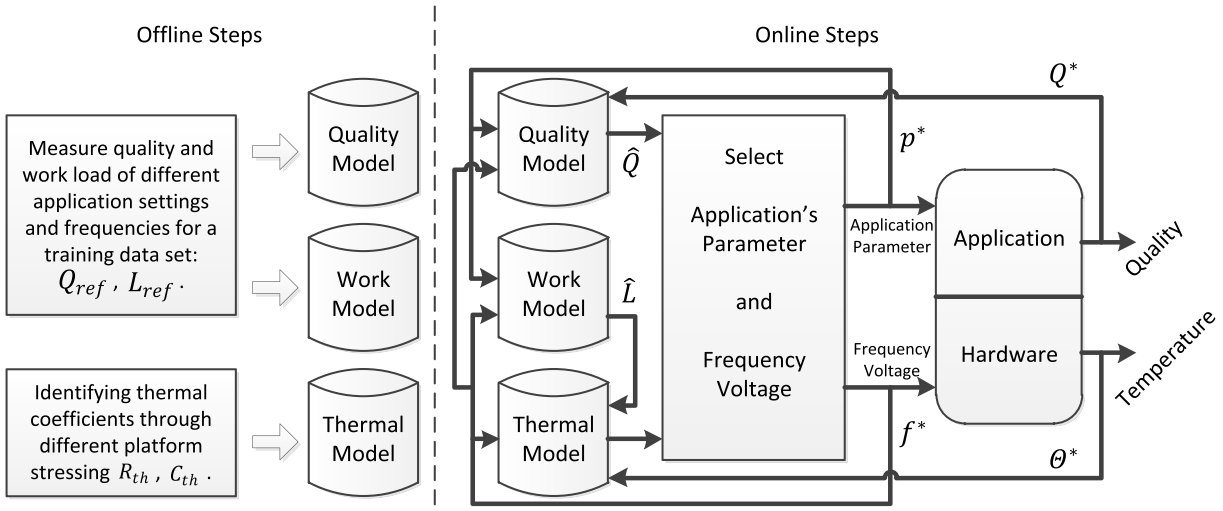

Fig. 9. Joint work and voltage/frequency scaling block diagram showing offline and online steps.

Lemma 2: For any point $(C_{c_0}, C_{r_0}) \in D_1$, on the contour line $\Theta(C_c, C_r) = \Theta_c$, there exist another point $(C_{c_1}, C_{r_1}) \in D_1$ on the contour line $\Theta(C_c, C_r) = \Theta_c$, which is closer to the D_1 boundary $C_r = C_c$ and $Q(C_{c_1}, C_{r_1}) > Q(C_{c_0}, C_{r_0})$.

Lemma 2 tells us for any point to be assumed as the solution of (14) in D_1, there will be another point with better quality, which contradicts the initial assumption. Therefore, the solution of (14) is not in D_1. To identify the solution in D_2, we first analyze applications with general contour lines of Fig. 7(a) where $\partial Q/\partial C_r > 0$ for all points in D_2.

Theorem 1: For a real-time application where $\partial Q/\partial C_r > 0$, the solution to the optimization problem (14) is the following point:

$$(C_c, C_r) : C_r = C_M \quad \text{and} \quad \Theta(C_c, C_M) = \Theta_c$$

The solution given in Theorem 1 can be achieved by superimposing the two contour lines plots of quality and temperature as shown in Fig. 8(a), where C_Q and C_Θ curves are quality and temperature contours, respectively. It is evident that $Q(C_{Q_3}) > Q(C_{Q_2}) > Q(C_{Q_1})$. However, none of the points on the contour line C_{Q_3} can be used for (14) solution due to lying in the red area and violating the thermal constraint. When walking across the quality contour lines from C_{Q_3} to C_{Q_1} the quality drops. Therefore, the solution of (14) can be found when the first quality contour line intersects with the critical temperature contour line. This occurs with C_{Q_2} contour line. As it is shown in the plot, the first point in which a quality contour line intersects with critical temperature contour line is when $C_r = C_M$. This point is the same solution point given in Theorem 1.

Theorem 2: For a real-time application where $\partial Q/\partial C_r$ changes sign at the points with $C_r = f(C_c)$, the solution to the optimization problem (14) is the following:

$$(C_c, C_r) : \Theta(C_c, f(C_c)) = \Theta_c \quad \text{and} \quad C_r = f(C_c)$$

Fig. 9(b) helps to visually understand the proof of Theorem 2.

Fig. 8(b) helps to visually understand the proof of Theorem 2. In fact, the visual proof of Theorem 2 follows

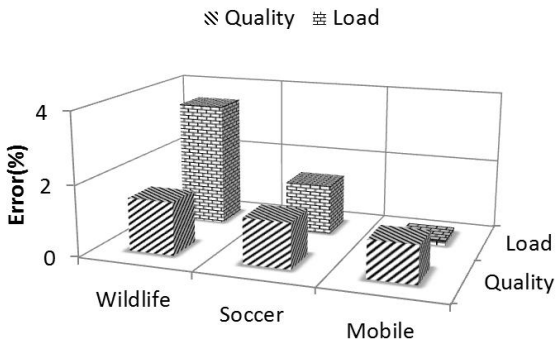

Fig. 10. Accuracy of functional quality and work load models for three test videos of Wild life, Soccer, and Mobile sequence.

the exact same steps of proof of Theorem 1. The solution is at the intersection of the first quality contour line with the critical temperature contour line, when walking across quality contour lines from C_{Q_3} toward C_{Q_1}. From Fig. 8(b), we see this point is located on the border of D_{2_1} and D_{2_2} which is $C_r = f(C_c)$.

In this section, we formulated DWVFS as a constrained optimization problem (14) and analytically characterized its solution. However, the derivation was performed under the assumption that quality and temperature, and the parameters they depend upon, are all continuous variables, which is not true in practice. Furthermore, even overlooking this assumption, (11) and (15) need to be solved numerically in real time to perform DTM decisions. Therefore, in the following section, we propose an efficient algorithm to solve the optimization problem described in (14).

V. Joint DWVFS

In this section, we introduce a fast and real time method for DWVFS based on the solution of optimization problem described in (14). Our method is a combination of offline steps and online steps. Fig. 9 shows the block diagram of the proposed method. In the offline steps, we collect the necessary data to develop three models. The first model captures the relationship of application's functional quality with respect to

the application parameter and the platform voltage/frequency. The second model describes the relationship between the application parameter and its work load. The last model is used to model platform's temperature with respect to the application's work load and platform's voltage/frequency. After a brief description of these models, we introduce the algorithm which finds the solution for (14), thereby maximizing application quality while performing thermal management.

A. Offline Models

1) Quality/Work Model: We model work load, $\hat{L}$, and functional quality, $\hat{Q}$, as functions of two variables, application parameter p and platform frequency f in the following way:

$$\begin{aligned} \mathbf{D} &= \{(p, f) | p \in \mathbf{P}, f \in \mathbf{F}\} \\ \hat{L} &: \mathbf{D} \rightarrow \mathbb{R}^+ \\ \hat{Q} &: \mathbf{D} \rightarrow \mathbb{R}^+. \end{aligned} \tag{16}$$

$\mathbf{P}$ is the set of possible values for the application parameter and $\mathbf{F}$ is the set of possible frequencies the platform can use.

We measure workload and functional quality of the application using a set of training data for all the points in $\mathbf{D}$. The results of these measurements are populated in two lookup tables (LUTs) used for reference

$$L_{\text{ref}}(p, f), Q_{\text{ref}}(p, f), \quad \text{where } (p, f) \in \mathbf{D}.$$

The model uses these reference LUTs and linearly scales them based on the measurements of the functional quality, Q^*, and work load, L^*, of the application during execution time

$$\begin{aligned} \hat{L}(p, f) &= \frac{L^*}{L_{\text{ref}}(p^*, f^*)} \cdot L_{\text{ref}}(pf) \\ \hat{Q}(p, f) &= \frac{Q^*}{Q_{\text{ref}}(p^*, f^*)} \cdot Q_{\text{ref}}(pf) \end{aligned} \tag{17}$$

where p^* and f^* are application parameter and platform frequency used in the last measurement of Q^* and L^*. Equation (17) estimates load ($\hat{L}$) and quality ($\hat{Q}$) for any p and f in $\mathbf{D}$ based on the measured L^* and Q^* for p^* and f^*. We validate the accuracy of the above models by applying them on the H.264 video encoder, whose quality function can be challenging to model as it is highly dependent on individual video streams and their content. The application parameter used is CRF and the functional quality metric is PSNR. We collected the reference LUT from a set of training video streams. Then, we applied the reference LUT for the modeling of three other test video streams. Fig. 10 shows the % error of the quality and load values estimated by the models from the actual measurements, which are only 4% and 2%, respectively, demonstrating accuracy of the models. Please note that the DWVFS algorithm proposed later is independent of how these models are developed.

2) Thermal Model: In this section, we describe our assumptions for thermal model that is incorporated in DWVFS. We discussed the thermal formula in (2) for a time period in which the power consumption is constant. Silicon power consumption is primarily composed of two components: dynamic, P_{dyn} and static P_{sta}

$$P = P_{\text{dyn}} + P_{\text{sta}}. \tag{18}$$

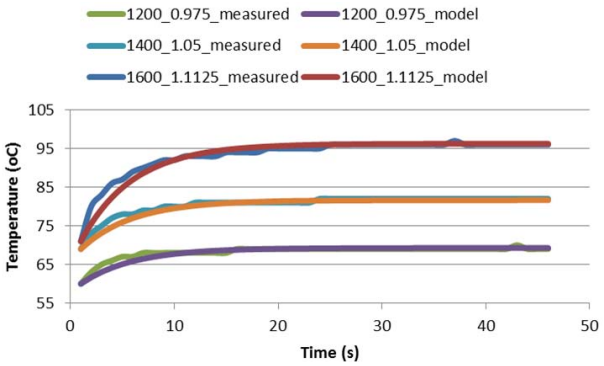

Fig. 11. Measured and modeled junction temperature of MacBook Air platform with Core2 Duo processor at 100% work load.

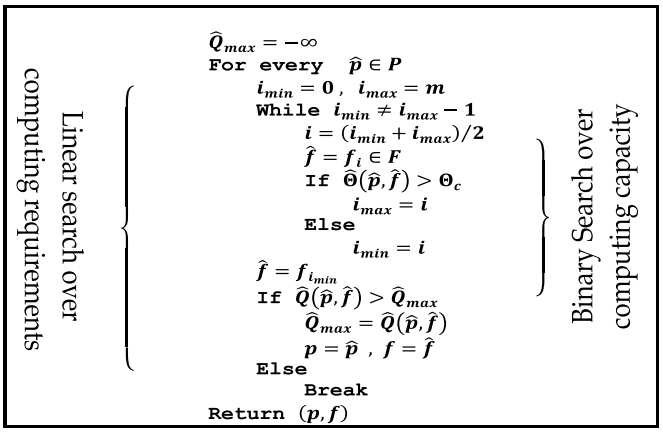

Fig. 12. Pseudocode of DWVFS algorithm (online steps).

TABLE I
VOLTAGE AND FREQUENCY LEVELS

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

The dynamic power is proportional to the hardware work load L as well as its frequency and voltage squared

$$P_{\text{dyn}} \propto V^2 f L. \tag{19}$$

The static power is proportional to the platform voltage and leakage current where the leakage current itself increases exponentially with increase in temperature [28]

$$P_{\text{sta}} \propto V I_{\text{leak}} \propto V e^{\alpha \Theta} \tag{20}$$

where α is the exponential proportionality coefficient and Θ is the temperature in degree Celsius. Therefore, based on (2) and (18)–(20), we get the following formula for estimating the junction temperature of the platform after a time interval of Δt with initial temperature of Θ^* and the ambient temperature of Θ_a:

$$\hat{\Theta}(\Delta t) = \Theta^* + \left(\Theta_a - \Theta^* + K_1 V^2 f L + K_2 V e^{\alpha \Theta^*}\right)\left(1 - e^{-\frac{\Delta t}{\tau}}\right). \tag{21}$$

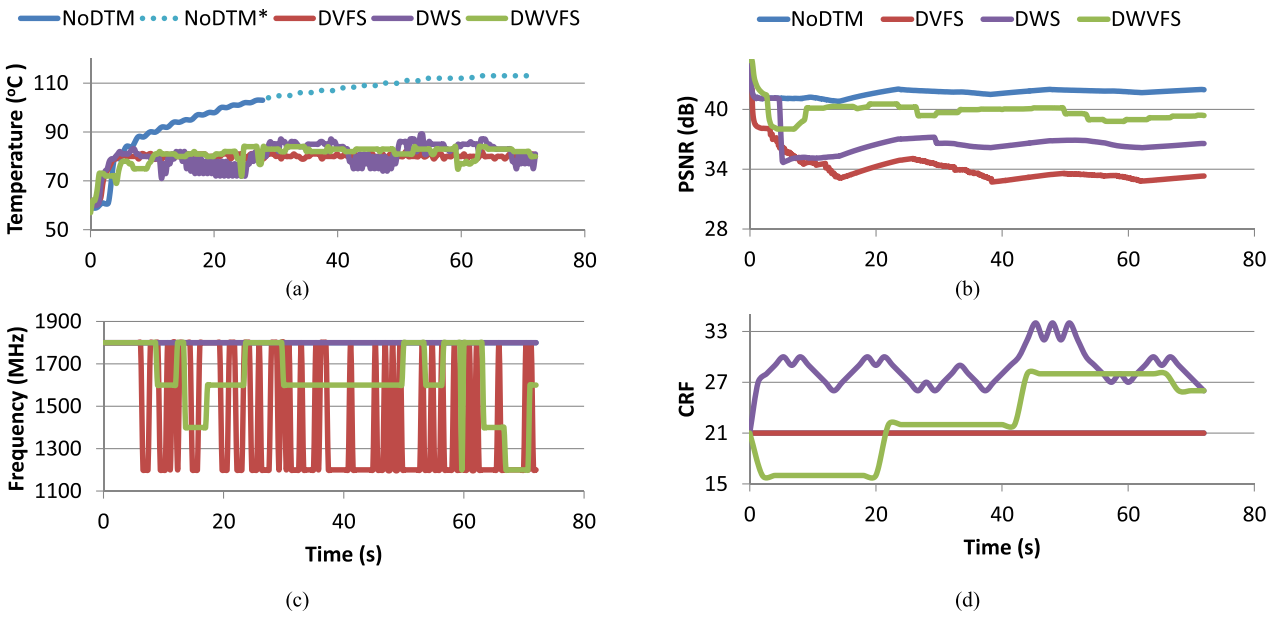

Fig. 13. Results of encoding wild life video clip with H.264 encoder using different thermal management algorithms. (a) Effects on the temperature. Due to thermal sensor limitation, it cannot report values more than 103 °C. NoDTM* plot is the projection of temperature as it rises. (b) Effects of different DTM methods on the encoding quality. (c) Frequency levels used during different thermal management algorithms. (d) CRF values used during different thermal management algorithms.

The model provided in (21) follows the same characteristics mentioned in Section IV for thermal contour lines (Fig. 6). When $C_r < C_c$ (L is variable) then the contour lines are decreasing and when $C_r > C_c$ (L is constant at 100%) the contour lines are vertical. The next step is to find the coefficients of the model described in (21) for a given platform. These coefficients are extracted for a MacBook Air (test platform) by stressing it in different conditions and using curve fitting methods. Fig. 11 shows the plots of the model provided in (21) and the measured temperature of this test platform in different operating conditions. Each curve on the plot corresponds to different frequency/voltages of the platform where these levels are predefined by manufacturer. The mean squared root errors of the model versus measurements are about 1 °C.

B. Online DWVFS Steps

Fig. 12 shows the pseudocode of the online steps for the joint DWVFS method which involves selecting the application's parameter and platform's voltage/frequency. The following paragraphs describe the proposed algorithm.

The algorithm selects the application parameter from a set $\mathbf{P} = \{p_1, p_2, \ldots, p_n\}$, sorted in decreasing order of computing requirement ($C_r(p_1) > C_r(p_2) > \cdots > C_r(p_n)$). The frequency is also selected from a set $\mathbf{F} = \{f_1, f_2, \ldots, f_m\}$ sorted in increasing order ($f_1 < f_2 < \cdots < f_m$). The outputs of the DWVFS algorithm are the new application parameter p, and platform's frequency f. The voltage of the platform is predefined for each frequency level.

The proposed algorithm is a combination of two nested searches in the directions of computing capacity and computing requirement (horizontal and vertical axes, respectively, in Fig. 8). We have mentioned that the temperature of the platform is an increasing function with respect to its computing capacity (6). We use this property and run a binary search in the direction of computing capacity over the temperature function, $\hat{\Theta}(C_c, C_r)$ as the inner search loop. In fact, the inner loop is based on Lemma 1 and the algorithm uses the thermal model to identify the points that are closest to the critical temperature contour line ($\hat{\Theta}(\hat{p}, \hat{f}) \leq \Theta_c$). Please note the computing capacity is represented by platform's frequency in Fig. 12.

Then, the selection process completes with a linear search in the direction of computing requirement over the quality function, $\mathrm{Q}(C_c, C_r)$, as the outer loop. Computing requirement is represented by the application parameter in Fig. 12. Since $\mathbf{P}$ is sorted in decreasing order of computing requirement, the outer loop sweeps the critical temperature contour line in the direction of point a to point b in Fig. 8. By using functional quality model, it compares the functional quality of the points on θ_c contour line, $(\hat{p}, \hat{f})$, and ends as soon as the quality drops. At the end, we have identified all the points on the critical temperature contour line (C_{θ_c} in Fig. 8) and selected the point which produces the maximum functional quality among them (C_{Q_2} and C_{θ_c} intersect in Fig. 8).

The proposed DWVFS algorithm is correct for both types of applications in Fig. 8. For the first type of application ($\partial Q/\partial C_r > 0$), Theorem 1 says the solution is the point on the θ_c contour line with the highest computing requirement. As we mentioned, this algorithm sweeps computing requirement in the decreasing order from point a to b. In the first iteration, it chooses the parameter with the highest C_r and the frequency which is located on θ_c contour line. In the next loop iteration, algorithm compares the next point of critical temperature contour line and since it has smaller quality than the first point (Theorem 1), the algorithm ends. For the second type of application, Theorem 2 claims that the functional quality

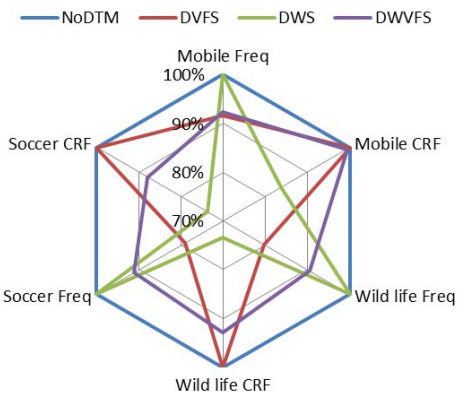

Fig. 14. Normalized average of the selected CRF and frequency values with different DTM and video streams.

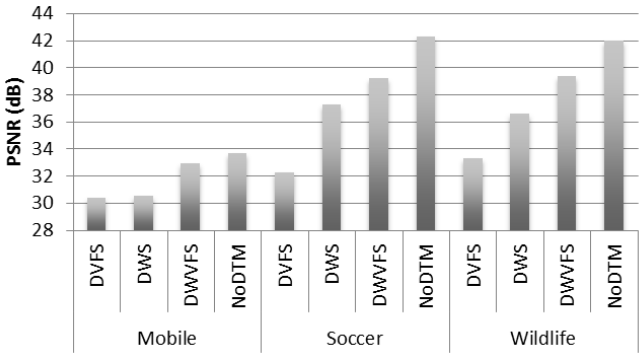

Fig. 15. Visual quality of the different video streams under different thermal managements.

Fig. 16. Block diagram of the turbo coding system.

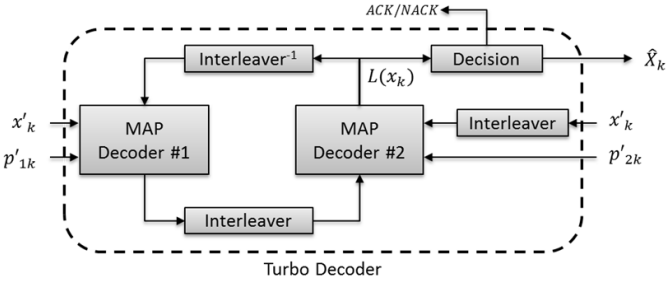

Fig. 17. Turbo decoder block diagram consisting of two MAP decoders.

increases from point a and somewhere in middle of θ_c contour line it drops. Again the algorithm captures the same behavior and finds the solution provided in Theorem 2. The above discussion shows that the DWVFS algorithm finds the solution for both types of applications and is correct by construction.

The proposed DWVFS algorithm has a time complexity of $O(n \log(m))$, where n is the number of values of application parameter, and m is the number of available frequency levels for the platform. Since the value of n is not large for typical applications (for, e.g., 11 and 10 for the H264 encoder and turbo decoder, respectively), and the value of m for typical platforms is small (like 5 for the MacBook Air platform), the algorithm can execute in real time. For example, the execution of the algorithm takes about 10–15 ms on the test platform depending on the CPU frequency; it is <0.04% of the decoding time for one H.264 frame. In the next section, we have used different DTM methods and shown the efficacy of DWVFS compared with other DTM methods.

VI. Experimental Results

In this section, we show how the proposed thermal management technique can be used on two different real-time applications, namely a video encoder and a turbo decoder. First, we briefly introduce the experimental setup that is used in this paper. Then, we discuss the results of applying the proposed thermal management algorithm on the video encoder and turbo decoder.

A. Experimental Setup

We evaluated the proposed thermal management technique using a MacBook Air laptop with an Intel Core2 Duo dual core processor, 2 GB of RAM, and a solid state disk. The operating system is Mac OS X 10.5. The laptop has a fan and a heat sink, which help with thermal control; however, they are not sufficient while running highly compute-intensive applications, as shown in the following section. The speed of the fan can reach up to 6200 rounds per minute. During all of our experiments, the fan operated at the maximum speed constantly.

The platform provides eight different frequency levels. The four main frequency levels are 1200, 1400, 1600, and 1800 MHz. In addition, by enabling an internal clock divider we can set platform frequency to 600, 700, 800, and 900 MHz. The selection of voltage level for each of these frequency levels are given by the vendor as specified in Table I.

For DVFS-based thermal management, we used the two frequency levels of 1200 and 1800 MHz. The reason for this selection is twofold. First, as mentioned in [10], effective thermal management is possible with only two frequency levels. In addition, the test results show that, having more number of frequencies reduces the application functional quality.

The temperature sensor in the platform reports the temperature with one degree Celsius accuracy up to 103°. If the temperature rises over 103°, it reports the constant value of 103. Therefore, for temperatures above 103, we extrapolated the curves in the plots.

B. H.264 Video Encoder

The first application used in this paper is an H.264 video encoder. We used an open source and highly efficient implementation of the H.264 standard called x264 [29]. The application parameter used for controlling video encoder's computing requirement is the CRF. The CRF is the default rate control method of x264 which tries to achieve the same perceptual quality throughout a video stream [33]. The CRF parameter can vary from 0 to 51, where a CRF of 0 produces a lossless compression and a CRF of 51 produces the lowest quality

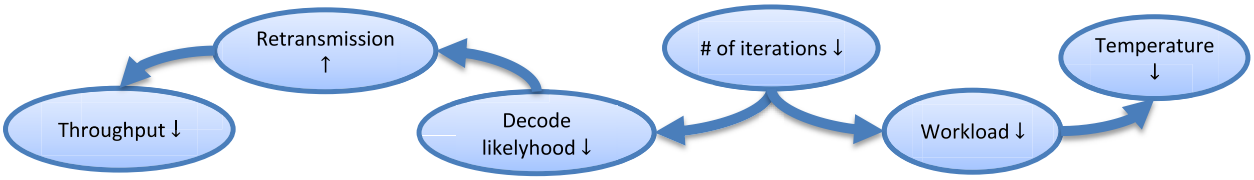

Fig. 18. Effect of changing the number of iterations in the turbo decoder on its throughput and temperature.

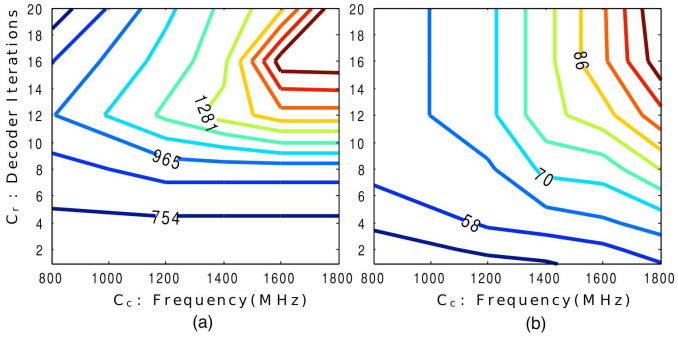

Fig. 19. Contour plots of the turbo decoder (a) functional quality measured as throughput in bps and (b) platform temperature.

result. We chose a range from 16 to 36 for CRF, with an increment of 2. It is evident that increasing CRF will decrease the computing requirement and vice versa.

To measure the functional quality, we used PSNR, which is one of the most widely used video quality metrics. To calculate PSNR, the original video stream is compared with the encoded video stream and the difference is measured in dB.

The video encoder's behavior is highly dependent on its input stream. For example, encoding a video stream with very little motion is less complex and easier than a high motion video stream. We used different video streams to capture this dependency. The first video stream is called wild life. Fig. 13 shows the results of encoding the wild life clip when different thermal management algorithms are used.

Fig. 13(a) shows the temperature versus time using different thermal management methods. As mentioned in Section VI-A, temperatures above 103 °C cannot be measured by the platform's temperature sensor. The dots in the plot are the projected values for temperature which cannot be read from the sensor. The critical temperature in this test is 80 °C. As shown in Fig. 13(a), all the DTM algorithms are able to control the temperature around the critical temperature. However, this comes with different penalties in the video quality. Fig. 13(b) shows the measured quality of the video stream over the duration of the encoding process. As we can see, DWVFS-based thermal management produces better quality compared with DWS and DVFS. In addition, DWS produces better quality compared with DVFS at all times. Fig. 13(c) and (d) shows the selection of frequency and CRF for different DTM algorithms. In the absence of DTM, the platform always runs with the maximum frequency and nominal CRF of 21. Comparing DWVFS with DWS and DVFS, we can see that DWVFS utilizes a combination of scaling frequency and CRF. Initially, DWVFS reduces the CRF (leading to higher quality) and as time passes it increases the CRF (leading to lower quality) for thermal management purposes. Concurrently, it also decreases the frequency in order to control the temperature while achieving higher quality. As shown in Fig. 13(c) and (d), the amount of increase in CRF in the case of DWVFS is not as much as in the case of DWS.

Since the complexity and quality of a video encoder is highly dependent on the video stream, we used three video clips to compare the thermal management techniques. Fig. 14 shows the normalized averages of the selected CRF and frequency values. Normalized average of selected frequency is the ratio of the average frequency used during a video encoder run to the nominal frequency of the platform (1.8 GHz). Since the CRF has inverse correlation with the video quality, we have used the ratio of the nominal CRF to the average of the selected CRF values as the normalized average CRF. As shown in Fig. 14, in the absence of any DTM algorithm, processor is always at its highest frequency and CRF is constantly at its nominal value. When DVFS is used, the CRFs for all the streams are still at the nominal values, but the average frequencies can go as low as 80% of the nominal frequency depending on the stream. On the other hand, the frequency is always at the nominal value when DWS is in use for all the streams but the average CRF used varies. The plot for DWVFS is always in between DVFS and DWS for both frequency and CRF. This means that DWVFS uses each control knob moderately to ensure that the drop in quality is minimized. Eventually, the different selection of CRF and frequency in the different DTM techniques leads into different stream quality, as shown in Fig. 15.

As shown in Fig. 15, the maximum quality drop due to the thermal management is about 10 dB. However, DWVFS helps to maintain a higher video quality and reduces the negative effects of thermal management. In addition, we should note that the high quality in the absence of DTM has been achieved with the price of increasing the processor's temperature over 110 °C. In practice, this will not happen if thermal protection is enabled, since a thermal management technique would be triggered when the platform reaches a maximum allowed temperature.

C. Turbo Decoder

The second application we used to verify the proposed thermal management scheme is a turbo decoder used in a turbo coding-based hybrid automatic repeat request (HARQ) [30] communication system. Fig. 16 shows the block diagram of the HARQ turbo coding system. The HARQ is a hybrid method that uses both forward error-correcting (FEC) codes and ARQ error controlling mechanism. In ARQ, the data are

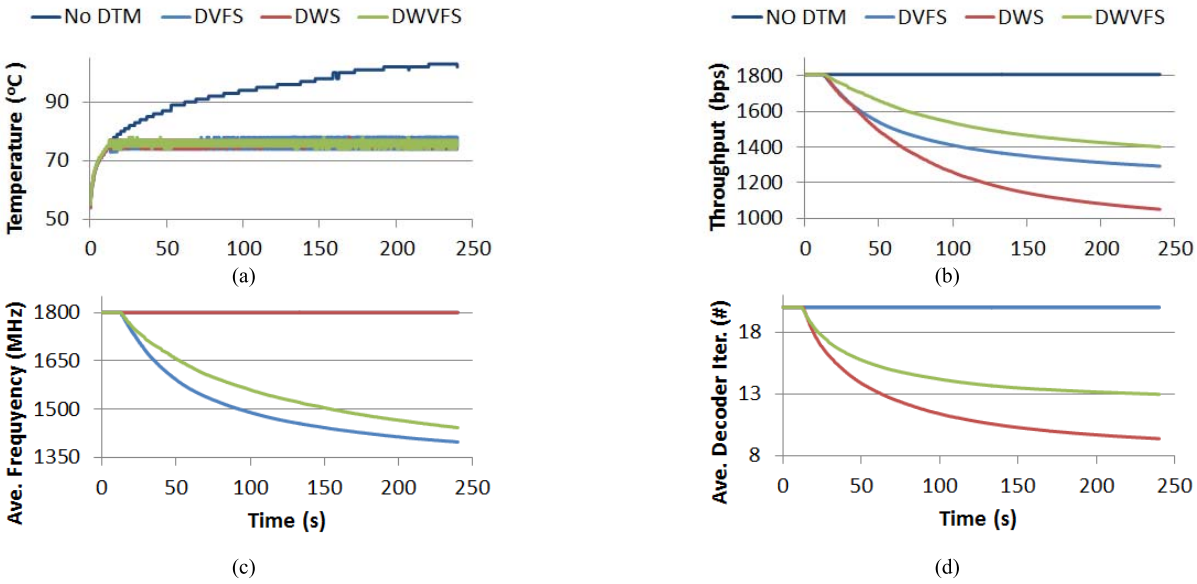

Fig. 20. Results of running the turbo decoder using different thermal management algorithms. (a) Effect on temperature. (b) Effect of different DTM methods on the turbo decoder throughput. (c) Average of frequency levels used by different thermal management algorithms. (d) Average number of decoder iterations by different thermal management algorithms.

sent with some error detecting (ED) code bits such as parity bits or cyclic redundancy check (CRC) bits [31]. When the receiver detects an error in the data stream using ED codes, it requests a retransmission. On the other hand, FEC codes help to not only find errors in the bit stream, but also help to correct the errors to some extent. In HARQ, the receiver first tries to detect and correct errors using the FEC codes. If the bit stream is not successfully retrieved using FEC codes, then the receiver requests retransmission. The receiver communicates with the sender through ACK or NACK messages (Fig. 16). X_k is the source information. The turbo encoder produces two series of parity bits: P_{1k} and P_{2k}. The information and parity bit symbols go through the channel and the noisy information and parity symbols (x'_k, p'_{1k}, p'_{2k}) are received by the decoder. We considered an additive white Gaussian noise channel with a noise distribution of $N(0, \sigma^2)$, with $\sigma = 0.95$. The decoded bits are denoted by $\hat{X}_k$.

In our experiments, we focus on the turbo decoder in the receiver side. Fig. 17 shows the block diagram of the turbo decoder. In this figure, $L(X_k)$ is the likelihood of finding the source information, X_k. The turbo decoder uses an iterative algorithm; every iteration of the loop increases the likelihood of finding the correct source information. Eventually, a decision will be made whether to request for a retransmission or the decoding is successfully completed.

The number of decoder iterations is the application parameter that is used in the DWS and DWVFS. The functional quality of turbo decoder application is its effective throughput. The effective throughput is the number of decoded bits, $\hat{X}_k$, per second. We have used the terms effective throughput and throughput interchangeably throughout this paper. When the platform gets too hot, by dropping the number of iterations, we reduce the work load and hence the platform temperature drops (Fig. 18). However, by reducing the number of iterations, the likelihood of correctly decoding source information decreases and we need more retransmissions. Higher number of retransmissions translates to a drop in the decoder's throughput. Therefore, the objective is to maximize the decoder's throughput while controlling the platform temperature. In this exercise, the frame size is 4701 bits and the interleaver is designed based on the UMTS standard [32].

Fig. 19 shows the quality versus temperature contour plots for the turbo decoder. The contour plots for the video encoder were presented earlier in Fig. 4. As shown in Figs. 4 and 19, the contour plots of the video encoder and turbo decoder follow the general shape that we discussed in Figs. 6 and 7.

Fig. 20 shows the results of running the turbo decoder with different thermal management algorithms on the MacBook Air platform. As shown in Fig. 20(a), the platform becomes very hot (over 100 °C) without thermal management which can deteriorate the reliability of the platform, necessitating the use of thermal management. Fig. 20(a) also shows that all the DTM methods control the temperature to the specified critical temperature of 75 °C. However, as we see in Fig. 20(b), different DTM methods affect the effective throughput of the decoder differently. Fig. 20(c) and (d) shows how the average values of frequency and application-level parameter (number of decoder iterations) vary across DTM algorithms. We see that in the case of DWVFS, the average frequency is higher than DVFS and average number of decoding iterations is higher than DWS. This suggests that DWVFS uses the best of both knobs to ensure that the functional quality, i.e., throughput, is maximized.

Fig. 21 compares the throughput of different DTM methods. In this scenario, the throughput of DVFS is better than DWS. However, DWVFS again achieves the best results compared with other DTM methods. According to this

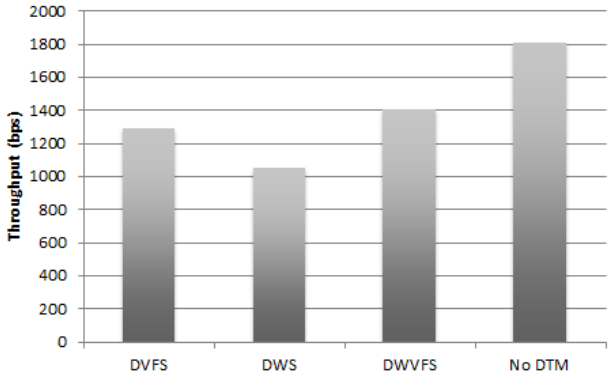

Fig. 21. Effects of different DTM methods on the throughput of the transmission.

figure, the throughput can drop by about 33% by using DWS but with the help of DWVFS the drop is limited to 22% only. In fact, DWVFS improves the throughput by 8% and 33% compared with DVFS- and DWS-based DTMs, respectively.

VII. Conclusion

In this paper, we showed how joint dynamic work/complexity scaling of real-time applications can be used as a new DTM method. We analyzed the effects of the platform's computing capacity and the application's computing requirements on both the platform's temperature and the application's functional quality. We showed how one can use these parameters to achieve a quality-optimized DTM. In addition, we formulated and analytically solved the optimization problem. We also proposed a fast algorithm to implement DWVFS.

In this paper, we only considered DVFS as the mechanism to change the computing capacity. However, there are other parameters that can be used to vary computing capacity, for example, the number of active cores. In addition, we only considered one parameter to scale the application's complexity.

In future work, the proposed approach can be extended to: 1) cover other mechanisms to vary computing capacity such as throttling number of CPU cores; 2) have multiple parameters affecting the workload of the application at the same time; and 3) scenarios when multiple real-time applications are running on the platform concurrently.

References

[1] D. Shin, S. W. Chung, E. Y. Chung, and N. Chang, "Energy-optimal dynamic thermal management: Computation and cooling power co-optimization," *IEEE Trans. Ind. Informat.*, vol. 6, no. 3, pp. 340–351, Aug. 2010.

[2] A. Kumar, L. Shang, L.-S. Peh, and N. K. Jha, "System-level dynamic thermal management for high-performance microprocessors," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 1, pp. 96–108, Jan. 2008.

[3] Y. Ge and Q. Qiu, "Dynamic thermal management for multimedia applications using machine learning," in *Proc. 48th ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Jun. 2011, pp. 95–100.

[4] E. Pakbaznia, M. Ghasemazar, and M. Pedram, "Temperature-aware dynamic resource provisioning in a power-optimized datacenter," in *Proc. Design, Autom. Test Eur. Conf. Exhibit. (DATE)*, Mar. 2010, pp. 124–129.

[5] W. Huang *et al.*, "TAPO: Thermal-aware power optimization techniques for servers and data centers," in *Proc. Int. Green Comput. Conf. Workshops (IGCC)*, Jul. 2011, pp. 1–8.

[6] K. Skadron, M. R. Stan, K. Sankaranarayanan, W. Huang, S. Velusamy, and D. Tarjan, "Temperature-aware microarchitecture: Modeling and implementation," *ACM Trans. Archit. Code Optim.*, vol. 1, no. 1, pp. 94–125, Mar. 2004.

[7] A. K. Coskun, "Efficient thermal management for multiprocessor systems," Ph.D. dissertation, Dept. Comput. Sci Eng., Univ. California, San Diego, CA, USA, 2009.

[8] V. Srinivasan, J. G. Hermerding, and R. Khanna, "An innovative approach to dynamic platform and thermal management for mobile platforms," in *Proc. Int. Conf. Energy Aware Comput. (ICEAC)*, Dec. 2010, pp. 1–4.

[9] S. Zhang and K. S. Chatha, "Thermal aware task sequencing on embedded processors," in *Proc. 47th ACM/IEEE Design Autom. Conf. (DAC)*, Jun. 2010, pp. 585–590.

[10] K. Skadron, "Hybrid architectural dynamic thermal management," in *Proc. Design, Autom. Test Eur. Conf. Exhibit.*, vol. 1. Feb. 2004, pp. 10–15.

[11] J. J. Chen, S. Wang, and L. Thiele, "Proactive speed scheduling for real-time tasks under thermal constraints," in *Proc. 15th IEEE Real-Time Embedded Technol. Appl. Symp. (RTAS)*, Apr. 2009, pp. 141–150.

[12] G. Quan, Y. Zhang, W. Wiles, and P. Pei, "Guaranteed scheduling for repetitive hard real-time tasks under the maximal temperature constraint," in *Proc. Int. Conf. Hardw./Softw. Codes. Syst. Synth. (CODES+ISSS)*, 2008, pp. 267–272.

[13] J. Park, H. M. Ustun, and J. A. Abraham, "Run-time prediction of the optimal performance point in DVS-based dynamic thermal management," in *Proc. 25th Int. Conf. VLSI Design (VLSID)*, Jan. 2012, pp. 155–160.

[14] X. Zhou, Y. Xu, Y. Du, Y. Zhang, and J. Yang, "Thermal management for 3D processors via task scheduling," in *Proc. 37th Int. Conf. Parallel Process. (ICPP)*, Sep. 2008, pp. 115–122.

[15] A. Kumar, L. Shang, L.-S. Peh, and N. K. Jha, "HybDTM: A coordinated hardware-software approach for dynamic thermal management," in *Proc. 43rd ACM/IEEE Design Autom. Conf.*, Jul. 2006, pp. 548–553.

[16] I. Yeo and E. J. Kim, "Hybrid dynamic thermal management based on statistical characteristics of multimedia applications," in *Proc. ACM/IEEE Int. Symp. Low Power Electron. Design (ISLPED)*, Aug. 2008, pp. 321–326.

[17] I. Yeo, H. K. Lee, E. J. Kim, and K. H. Yum, "Effective dynamic thermal management for MPEG-4 decoding," in *Proc. 25th Int. Conf. Comput. Design (ICCD)*, Oct. 2007, pp. 623–628.

[18] W. Lee, K. Patel, and M. Pedram, "GOP-level dynamic thermal management in MPEG-2 decoding," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 16, no. 6, pp. 662–672, Jun. 2008.

[19] V. Hanumaiah and S. Vrudhula, "Temperature-aware DVFS for hard real-time applications on multi-core processors," *IEEE Trans. Comput.*, vol. 61, no. 10, pp. 1484–1494, Oct. 2012.

[20] D. Forte and A. Srivastava, "Energy and thermal-aware video coding via encoder/decoder workload balancing," in *Proc. ACM/IEEE Int. Symp. Low-Power Electron. Design (ISLPED)*, Aug. 2010, pp. 207–212.

[21] W. Lee, K. Patel, and M. Pedram, "Dynamic thermal management for MPEG-2 decoding," in *Proc. Int. Symp. Low Power Electron. Design (ISLPED)*, Oct. 2006, pp. 316–321.

[22] A. Pant, P. Gupta, and M. van der Schaar, "AppAdapt: Opportunistic application adaptation in presence of hardware variation," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 11, pp. 1986–1996, Nov. 2012.

[23] S. Wang and S. Dey, "Rendering adaptation to address communication and computation constraints in cloud mobile gaming," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM)*, Dec. 2010, pp. 1–6.

[24] C. Schlegel and L. Perez, *Trellis and Turbo Coding* (Digital & Mobile Communication), J. B. Anderson, Ed. New York, NY, USA: Wiley, 2004.

[25] Y. Lin, S. Mahlke, T. Mudge, C. Chakrabarti, A. Reid, and K. Flautner, "Design and implementation of turbo decoders for software defined radio," in *Proc. IEEE Workshop Signal Process. Syst. Design Implement. (SIPS)*, Oct. 2006, pp. 22–27.

[26] K. Skadron, T. Abdelzaher, and M. R. Stan, "Control-theoretic techniques and thermal-RC modeling for accurate and localized dynamic thermal management," in *Proc. 8th Int. Symp. High-Perform. Comput. Archit.*, Feb. 2002, pp. 17–28.

[27] R. Courant, H. Robbins, and I. Stewart, *What Is Mathematics? An Elementary Approach to Ideas and Methods*. New York, NY, USA: Oxford Univ. Press, 1996, p. 344.
[28] J. A. Butts and G. S. Sohi, "A static power model for architects," in *Proc. 33rd Annu. IEEE/ACM Int. Symp. Microarchit. (MICRO)*, Dec. 2000, pp. 191–201.
[29] x264. (2014, Jul. 17). *VideoLan* [Online]. Available: http://www.videolan.org/developers/x264.html
[30] W. Yafeng, Z. Lei, and Y. Dacheng, "Performance analysis of type III HARQ with turbo codes," in *Proc. 57th IEEE Semiannu. Veh. Technol. Conf.*, vol. 4. Apr. 2003, pp. 2740–2744.
[31] M. Stigge, H. Plötz, W. Müller, and J. P. Redlich, *Reversing CRC—Theory and Practice*. Berlin, Germany: Humboldt University Berlin, 2011, p. 17.
[32] *Universal Mobile Telecommunications System (UMTS). Multiplexing and Channel Coding (FDD) (3G TS 25.212 Version 3.1. 1 Release 1999)*, document ETSI, TS. 125 212 (V3. 1.1), 2000.
[33] MeWiki. (2014, Jul. 17). *X264 Settings* [Online]. Available: http://mewiki.project357.com/wiki/X264_Settings#crf
[34] H. Huang, G. Quan, J. Fan, and M. Qiu, "Throughput maximization for periodic real-time systems under the maximal temperature constraint," in *Proc. 48th ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Jun. 2011, pp. 363–368.
[35] S. Wang and R. Bettati, "Reactive speed control in temperature-constrained real-time systems," in *Proc. Euromicro Conf. Real-Time Syst.*, 2006.
[36] R. Rao and S. Vrudhula, "Performance optimal processor throttling under thermal constraints," in *Proc. Int. Conf. Compilers, Archit., Synth. Embedded Syst. (CASES)*, 2007, pp. 257–266.
[37] G. Wu and Z. Xu, "Temperature-aware task scheduling algorithm for soft real-time multi-core systems," *J. Syst. Softw.*, vol. 83, no. 12, pp. 2579–2590, Dec. 2010.
[38] H. Yang, I. Bacivarov, D. Rai, J. Chen, and L. Thiele, "Real-time worst-case temperature analysis with temperature-dependent parameters," *J. Real-Time Syst.*, vol. 49, no. 6, pp. 730–762, 2013.

Ali Mirtar received the B.Sc. and M.Sc. degrees in electrical engineering from the Sharif University of Technology, Tehran, Iran, in 2006 and 2008, respectively. He is currently pursuing the Ph.D. degree in computer engineering with the Mobile Systems Design Laboratory, University of California at San Diego, La Jolla, CA, USA.

He joined the Mobile and Wireless Group, Broadcom, Irvine, CA, USA, in 2012, as a Systems Designer, where he was involved in next-generation cellphone's power and thermal optimization. He is currently with Broadcom. He has published several conference and journal papers, and holds two U.S. and Iran patents.

Sujit Dey (SM'03–F'13) received the Ph.D. degree in computer science from Duke University, Durham, NC, USA, in 1991.

He served as the Chief Scientist, Mobile Networks, with Allot Communications Ltd., Hod HaSharon, Israel, from 2012 to 2013. He founded Ortiva Wireless Inc., La Jolla, CA, USA, in 2004, where he served as its founding CEO and later as a CTO till its acquisition by Allot Communications in 2012. Prior to Ortiva, he served as the Chair of the Advisory Board of Zyray Wireless Inc., San Diego, CA, USA, till its acquisition by Broadcom in 2004. Prior to joining the University of California at San Diego (UCSD), La Jolla, in 1997, he was a Senior Research Staff Member with the NEC Research Laboratories, Princeton, NJ, USA. He is a Professor with the Department of Electrical and Computer Engineering, UCSD, where he heads the Mobile Systems Design Laboratory, which is involved in developing innovative mobile cloud computing architectures and algorithms, adaptive multimedia and networking techniques, low-energy computing and communication, and reliable system-on-chips, to enable the next generation of mobile multimedia applications. He is affiliated with the Qualcomm Institute and the Center for Wireless Communications at UCSD. He has co-authored more than 200 publications, including journal and conference papers, and a book on low-power design. He is the co-inventor of 17 U.S. and two international patents, resulting in multiple technology licensing and commercialization.

Dr. Dey was a recipient of six IEEE/ACM Best Paper Awards, and has chaired multiple IEEE conferences and workshops.

Anand Raghunathan received the B.Tech. degree in electrical and electronics engineering from IIT Madras, Chennai, India, and the M.A. and Ph.D. degrees in electrical engineering from Princeton University, Princeton, NJ, USA.

He was a Senior Research Staff Member with NEC Laboratories America Inc., Princeton, NJ, USA, and was the Gopalakrishnan Visiting Chair with the Department of Computer Science and Engineering, IIT Madras, Chennai, India. He is a Professor with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA, where he leads the Integrated Systems Laboratory. He has co-authored a book *High-Level Power Analysis and Optimization*, eight book chapters, over 200 refereed journal and conference papers, and holds 21 U.S. patents. His current research interests include domain-specific architecture, system-on-chip design, embedded systems, and heterogeneous parallel computing.

Prof. Raghunathan has served on the technical program and organizing committees of several leading conferences and workshops. He has chaired the ACM/IEEE International Symposium on Low Power Electronics and Design, the ACM/IEEE International Conference on Compilers, Architecture, and Synthesis for Embedded Systems, the IEEE VLSI Test Symposium, and the IEEE International Conference on VLSI Design. He has served as an Associate Editor of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN, the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION SYSTEMS, the *ACM Transactions on Design Automation of Electronic Systems*, the IEEE TRANSACTIONS ON MOBILE COMPUTING, the *ACM Transactions on Embedded Computing Systems*, the IEEE DESIGN & TEST OF COMPUTERS, and the *Journal of Low Power Electronics*. He was a recipient of the IEEE Meritorious Service Award in 2001 and the Outstanding Service Award in 2004. He is a Golden Core Member of the IEEE Computer Society. His publications have been recognized with eight Best Paper Awards and four Best Paper Nominations. He received the Patent of the Year Award (recognizing the invention with the highest impact), and two Technology Commercialization Awards from NEC. He was chosen by MIT's Technology Review among the TR35 (top 35 innovators under 35 years, across various disciplines of science and technology) in 2006, for his work on making mobile secure.