Cool Shapers: Shaping Real-Time Tasks for Improved Thermal Guarantees

Pratyush Kumar and Lothar Thiele
Computer Engineering and Networks Laboratory, ETH Zürich
{pratyush.kumar, lothar.thiele}@tik.ee.ethz.ch

ABSTRACT

With increasing power densities, managing on-chip temperatures has become an important design challenge. We propose a novel approach to this problem with the use of shapers to dynamically and selectively insert idle times during the execution of hard real-time jobs on a single speed processor. For the class of leaky bucket shapers which have a light-weight implementation, we derive the shaper such that no job misses its real-time deadline and the peak temperature is optimally reduced. The analysis and design of such shapers allows for dynamically variable streams of jobs; for instance, periodic streams with jitter. We extend our results to consider non-zero power and timing overhead in transitioning to the idle mode. With experimental results, we demonstrate that the proposed approach provides a large improvement: on average 8K peak temperature reduction or 40% increase in utilization for a given peak temperature.

Categories and Subject Descriptors

D.4.7 [**Software**]: Operating systems—*Real-time systems and embedded systems*; B.8.0 [**Hardware**]: Performance and Reliability—*General*

General Terms

Design, Theory

Keywords

Real-Time Systems, Thermal Management, Shapers

1. INTRODUCTION

Power densities in electronic devices have risen dramatically translating to high on-chip temperatures. High temperatures not only raise long-term reliability concerns, but can also affect the functional correctness of the system. Furthermore, at higher temperatures the leakage power is higher, leading to a vicious cycle that can cause thermal run-aways. As a consequence, thermal management has become a key system design issue.

Hardware cooling solutions remain the front-line approach to remove heat from electronic devices. However, the cost of packaging technology is increasing exponentially and is currently estimated to cost $3 per watt of heat dissipated [8]. It is now common practice to supplement such hardware solutions with run-time methods which are broadly classified as Dynamic Thermal Management (DTM) techniques [4]. DTM techniques aim to manage on-chip temperatures by using actuators such as voltage and/or frequency scaling, clock gating, execution unit throttling and so on.

Clearly, while employing such DTM techniques, the performance penalties have to be considered. This is especially significant for hard real-time applications, where it must be ensured that jobs complete within their respective deadlines. This trade-off introduces interesting problems, and consequently, thermal-aware real-time scheduling has been an active line of research. Bansal et al. [3] have studied the problem of maximizing the total resource that can be supported by a system under specified temperature constraints. In essence, the authors assume a never-empty buffer of waiting jobs with characterized execution times, and question how many of those jobs can be completed within the considered time. Other researches have studied the problem of scheduling a statically specified task-set. Zhang et al. [13] provide approximation algorithms to minimize the completion time of a set of periodic tasks under restriction of single speed per task. Chen et al. [5] study the problem of speed control for a given trace of periodically repeating tasks. These and similar other studies are restricted in that they consider a *given* trace of job (individual instances of a task) arrivals. In reality, such static characterization is not available: input streams exhibit occasional and variable bursts, the specification and analysis of which is crucial for providing real-time guarantees. Consequently, conventional real-time analysis is performed with general job arrival patterns that are modelled in frameworks such as Real-Time Calculus (RTC) [9]. In view of this, a good DTM technique designed for real-time applications must have the following properties: (a) works with general job arrival patterns, (b) guarantees real-time constraints of all jobs, (b) minimizes the peak temperature, and (d) is lightweight to implement.

In this paper, we present a novel DTM technique for a single speed processor that satisfies all the above properties. To the best of our knowledge, this is the first such technique that considers general job arrival patterns. In our approach, temperature is managed by using leaky bucket shapers to dy-

DAC 2011, June 5-10, 2011, San Diego, California, USA.

namically and selectively put the processor in a low power idle mode, while ensuring that no job misses its real-time deadline. The organization and main results of the paper are as follows. Within the class of leaky bucket shapers, we analytically compute the shaper that optimally minimizes the maximum temperature (Section 4). We modify the optimal shaper for systems with non-zero power and timing overhead incurred while transitioning to the idle mode (Section 5). We extend these results to consider systems where multiple tasks are scheduled with the Earliest Deadline First (EDF) policy (Section 6). We present the implementation of the proposed shapers which is light-weight and has minimal run-time requirements (Section 7). Finally using experimental results we show that large reductions in the maximum temperature (on average 8K) and large increases in supported utilization (on average 40%) are obtained by the proposed technique (Section 8).

2. SYSTEM MODELS

2.1 Processor model

We consider a system, where the processor executes jobs at a fixed frequency f and consumes power P_{act}. When not executing jobs, the processor consumes a lower power P_{idl}. The processor is said to be in active and idle modes when executing and not executing jobs, respectively. Under normal operation, transition between modes happens automatically, depending on the task-queue. Additionally, independent of the state of the task-queue, the processor can be *forced* to remain in the idle mode by techniques such as execution throttling [4] or fetch toggling [8]. With such a technique, a transition overhead of t_{tr} time units is required to move from the active mode to the idle mode and back to the active mode. During this transition time, the processor consumes power P_{act} but does not execute any jobs.

With increasing dominance of leakage power in modern VLSI systems, the power consumption depends on the temperature. As detailed in [7], a linear model works well in practice: $P = \rho T + \omega$. These parameters are different for either mode of the processor and we reference the mode by sub-scripts *act* and *idl*. Thus, the processor is characterized by the tuple $\mathbf{P} = (f, \rho_{act}, \omega_{act}, \rho_{idl}, \omega_{idl}, t_{tr})$.

2.2 Task model

We model a task as a stream of jobs characterized by a *resource-based upper arrival curve* [9] denoted as α. Let the resource demand of jobs that arrive between $[0, t)$ be $W(t)$, i.e., the sum of execution times of jobs arriving before time t equals $W(t)$. Then, $\alpha(\Delta)$ upper-bounds the resource demand of all jobs that arrive in any interval of length Δ,

$$W(t + \Delta) - W(t) \leq \alpha(\Delta). \quad (1)$$

Arrival curves substantially generalize simple models of job arrivals such as periodic and sporadic. Indeed, they have been extensively used to analyze complex real-time systems [9]. In addition to the arrival curve, the stream is characterized by a relative deadline D, i.e., each job must complete execution within D time units of its arrival. Thus, the task model is characterized by the tuple $\tau = (\alpha, D)$.

2.3 Thermal model

For studies in architectural-level thermal management [3, 5, 13], heat flow is approximated by using the Fourier's Law assuming that the processor is a point source. For such an approximation, the temperature of the processor, T, evolves according to the following differential equation

$$C\frac{dT}{dt} = -G(T - T_{amb}) + P, \quad (2)$$

where C and G are thermal model parameters, T_{amb} is the ambient temperature, and P is the power consumed by the processor. For the considered linear power model, the closed form solution to (2) for a starting time t_0 is given as

$$T(t) = T^{\infty} + (T(t_0) - T^{\infty}) \cdot e^{-a(t-t_0)}, \quad (3)$$

where $T^{\infty} = \frac{GT_{amb}+\omega}{G-\rho}$, and $a = \frac{G-\rho}{C}$. We again reference these parameters with sub-scripts to represent the mode of the processor. Thus, the thermal model of the system is characterized by the tuple $\mathbf{T} = (T^{\infty}_{act}, a_{act}, T^{\infty}_{idl}, a_{idl})$.

2.4 Problem definition

Given are a processor $\mathbf{P}$, a task model τ, and a thermal model $\mathbf{T}$. We are to design a DTM policy, that should at all times choose the mode of the processor (active or idle) such that (a) all jobs complete within their deadlines, and (b) the peak temperature of the system is minimized.

The DTM policy can reduce the temperature by putting the processor in the idle mode. However, this has to be carefully done. If the processor is kept in the idle mode for too long, a burst of jobs may arrive and lead to deadline misses. This is critical for general job arrival patterns, where the time of arrival of bursts is unknown. On the other hand, if the processor is switched too often between active and idle modes, a large overhead may be incurred during transition, leading to higher temperatures. Thus, when and for long to put the processor in the idle mode are crucial decisions.

3. PRELIMINARIES

We start by presenting some known results from Real-Time Calculus (RTC) that we shall use later in the paper. Consider a stream of jobs with an arrival curve α, as defined in Section 2.2. These jobs are to be executed on a workload conserving processor, i.e., an always-available processor.

3.1 Schedulability analysis

From [9], the maximum delay d_{max} experienced by any job in the stream can be upper-bounded as

$$d_{max} \leq \sup_{\lambda \geq 0} \{\inf\{\tau \geq 0 : \alpha(\lambda) \leq \lambda + \tau\}\} \stackrel{def}{=} Del(\alpha, \mathbf{1}). \quad (4)$$

The above bound is simply the largest horizontal distance between the α curve and the line passing through origin with slope 1. For a given relative deadline, D, (4) can be formulated as a *schedulability constraint* as

$$\alpha(\Delta - D) \leq \Delta. \quad (5)$$

3.2 Peak temperature analysis

In [1], a method is proposed to compute the peak temperature of a workload conserving processor with a thermal model as described in Section 2.3, when executing a stream of jobs characterized by an upper arrival curve. The following theorem follows from the results of [1].

THEOREM 1. *Given are two tasks τ_1 and τ_2 with resource-based upper arrival curves α_1 and α_2, respectively, with $\alpha_1 \leq \alpha_2$. Then the peak temperature of a workload-conserving processor when serving τ_2 is higher than that when serving τ_1.*

It must be noted that both the above analyses do not consider any DTM policy enabled on the processor. We shall see later, how to adapt these results in the presence of a DTM policy based on shaping of the jobs.

4. OPTIMAL LEAKY-BUCKET SHAPER

In this section, we assume that transition between the two modes of the processor does not have any timing or power overhead. For this simplified setting, we obtain the optimal shaper within the class of leaky bucket shapers. In Section 5, we consider a non-zero transition overhead.

4.1 DTM with shapers

In the area of networking, traffic shaping is a well-studied technique to regulate flow of packets by buffering and delaying them [6]. For instance, a shaper may ensure that the output stream has limited burstiness and thereby reduce the buffer requirement in a downstream node.

The temperature of the processor depends on how *long* the processor executes rather than how *many* jobs it executes. Hence, rather than the conventional approach of shaping based on the number of jobs (or packets), we study shapers that shape the resource demand of jobs. Using such shapers for DTM in real-time systems has some crucial advantages. Firstly, shapers work dynamically and respond to the time-varying resource demand of arriving jobs. Further, well-designed shapers delay execution only when such delay does not violate the deadline of any job. For a given system, several such shapers, which guarantee deadlines can exist. The problem then is to use this degree of freedom, in the choice of shapers, to minimize the highest temperature.

4.2 Greedy shapers

A greedy shaper is an abstract RTC component [10] which models a shaper, but additionally ensures that no job is delayed more than it needs to be. It is characterized by a *resource-based shaping curve* denoted as σ. When a stream of jobs is shaped by a shaping curve σ, the resource-based upper arrival curve, α_s, of jobs at the output of the shaper is given by the RTC convolution operator $\otimes$ [9] as

$$\alpha_s = \alpha \otimes \sigma \stackrel{def}{=} \inf_{0 \le \lambda \le \Delta} \{\alpha(\Delta - \lambda) + \sigma(\lambda)\} \quad (6)$$

As a property of the convolution operator, we have $\alpha_s \le \sigma$. Equivalently expressed in terms of the resource demand of jobs that appear at the output of the shaper between $[0, t)$, denoted as $W_s(t)$, we have

$$W_s(t + \Delta) - W_s(t) \le \sigma(\Delta), \quad \forall\, t > 0. \quad (7)$$

The maximum amount of time by which the shaper delays any job denoted as d^s_{max} is upper-bounded as

$$d^s_{max} \le \sup_{\lambda \ge 0}\{\inf\{\tau \ge 0 : \alpha(\lambda) \le \sigma(\lambda + \tau)\}\} = Del(\alpha, \sigma) \quad (8)$$

4.3 Leaky bucket shapers

Implementing a general greedy shaper with arbitrarily complex shaping curve, σ, can be resource-heavy. For every $\Delta_1 > 0$, it must be ensured that the amount of resource demanded in the last Δ_1 units of time does not exceed $\sigma(\Delta_1)$. This requires large storage space and computation resources. Leaky bucket shapers are a special class of greedy shapers with efficient implementation. A leaky bucket shaper with bucket size b_i, a fill rate r_i and infinitesimally small resource demand granularity has a shaping curve σ_i given as

$$\sigma_i(\Delta) = b_i + r_i \Delta \quad (9)$$

A cascade of such leaky bucket shapers can be used to obtain a greedy shaper with a shaping curve

$$\sigma(\Delta) = \min_{\forall i}\{b_i + r_i \Delta\} \quad (10)$$

As we will see later in Section 7, lightweight implementation of such shapers can be designed.

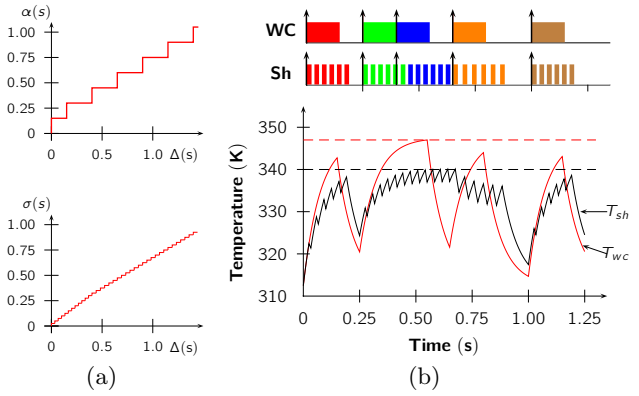

Figure 1: Motivating example

4.4 Motivating example

We illustrate with an example, temperature management using leaky bucket shapers. Consider a stream of jobs with a period $P = 0.25$s, jitter $J = 0.1$s, worst-case execution time (WCET) $C = 0.15$s and relative deadline $D = 0.25$s. The arrival curve α for this stream is shown in Figure 1(a). The power and thermal models are as described later in Table 1. Consider a specific trace of jobs that arrive at times $t = (0, 0.25, 0.4, 0.65, 1)$s. We study the execution of the jobs under two policies: (a) a workload conserving execution (`WC`) with no DTM policy enabled, and (b) a shaped execution (`Sh`) where the jobs are delayed according to a specifically designed shaping curve σ shown in Figure 1(a). For either policy, the evolution of the temperature of the processor over time is shown in Figure 1(b).

The peak temperature in trace `Sh` is 7K (=347-340) less than that in trace `WC`. In `Sh`, the shaper dynamically delays the execution of the jobs by putting the processor in the idle mode. This delays the completion time of jobs, when compared to the trace `WC`. However, for the designed shaping curve, all jobs complete within their deadlines. When job 3 arrives at time $t = 0.4$s, leading to a temporary burst, the shaping ensures that jobs 2 and 3 complete within their deadlines. Having crossed the burst, job 4 is shaped to run with more idle time in between the executions, demonstrating the dynamic nature of the shaper.

4.5 Analysis with shapers

Having demonstrated the advantage of shaping, we now discuss schedulability and peak temperature analyses in the presence of shaping. Let α and σ denote the resource-based upper arrival curve of the stream and the shaping curve, respectively. Then, we have the following theorems.

THEOREM 2. *The total delay between the arrival of a job at the shaper and the completion of its execution at the processor is upper-bounded by d^{tot}_{max}, which is given as*

$$d^{tot}_{max} = Del(\alpha, \sigma) + Del(\sigma, \mathbf{1}) \quad (11)$$

PROOF. Jobs pass through the shaper and then are executed on the processor. So, the total delay can be upper-bounded by the sum of the maximum delays in each component. The maximum delay of a job through the shaper is obtained directly from (8). The maximum delay of a job in being served by the processor is obtained by applying (4) as $Del(\alpha_s, \mathbf{1})$, where α_s denotes the resource-based upper ar-

rival curve at the output of the shaper. But we know from (6) that $\alpha_s \leq \sigma$ and thus $Del(\alpha_s, \mathbf{1}) \leq Del(\sigma, \mathbf{1})$. □

THEOREM 3. *Given is a task τ with an arrival curve α. Consider two shaping curves σ and σ' such that $\sigma' \leq \sigma$. Then the peak temperature when shaping tasks with σ' cannot be larger than that when shaping tasks with σ.*

PROOF. Since $\sigma' \leq \sigma$, from (6) we have $\alpha \otimes \sigma' \leq \alpha \otimes \sigma \Rightarrow \alpha'_s \leq \alpha_s$. From Theorem 1, the peak temperature when serving a task with arrival curve α'_s cannot be higher than when serving a task with arrival curve α_s. □

We are now ready to design the optimal shaper. Define $ConvexHull(\alpha)$ as the upper convex hull of the α curve.

THEOREM 4. *Given are a processor $\mathbf{P}$ and a task model τ. Amongst all leaky bucket shapers that satisfy the schedulability constraint, the leaky bucket shaper with a shaping curve $\sigma_{ch} = ConvexHull(\alpha(\Delta - D))$ optimally reduces the highest temperature of the processor.*

PROOF. First we show that the schedulability constraint holds. Since, σ_{ch} is the upper convex hull of $\alpha(\Delta - D)$, we have $\sigma_{ch} \geq \alpha(\Delta - D)$. Thus, $Del(\alpha, \sigma_{ch}) = D$. Now to show that $d^{tot}_{max} \leq D$, from Theorem 2, we need to only show that $Del(\sigma_{ch}, \mathbf{1}) = 0$, or equivalently, $\sigma_{ch}(\Delta) \leq \Delta$. We show this by contradiction. Let Δ_1 be the largest number such that $\sigma_{ch}(\Delta_1) \geq \Delta_1$. Then, we have $\alpha(\Delta_1 - D) \geq \Delta_1$ and hence $Del(\alpha, \mathbf{1}) > D$. This implies that even for a workload conserving execution, the processor is not capable of completing all jobs within their deadlines. We restrict ourselves to problem instances that are *feasible*. For such problem instances, the desired condition $\sigma_{ch}(\Delta) \leq \Delta$ holds.

Now we need to show that there exists no other leaky bucket shaper that satisfies the schedulability constraint and has a lower highest temperature. We prove this by contradiction. Let such a shaper exist with a shaping curve σ'. Since, it satisfies schedulability constraint, we have from Theorem 2, $\sigma' \geq \alpha(\Delta - D)$. Further, since shaping with σ' leads to a lower peak temperature than with σ_{ch}, from Theorem 3, we have $\sigma_{ch} \not\leq \sigma'$. Let $\sigma'' = \min\{\sigma_{ch}, \sigma'\}$. Clearly, $\sigma'' < \sigma_{ch}$. The shaping curve of a cascade of leaky buckets bounds a convex region as it is the minimum of a set of lines (10). With both σ_{ch} and σ' bounding a convex region and greater than $\alpha(\Delta - D)$, σ'' also bounds convex region and is greater than $\alpha(\Delta - D)$. This contradicts the definition of σ_{ch} as the upper convex hull of $\alpha(\Delta - D)$. □

Importantly, the optimal leaky bucket shaper does not depend on the power or thermal properties of the processor. It is fully characterized by the task model: $\tau = (\alpha, D)$.

5. NON-ZERO TRANSITION OVERHEAD

In the previous section, we assumed no overhead in transitioning between modes. We thus allowed the leaky bucket shapers to shape the jobs with infinitesimally small resource demand granularity; thereby theoretically switching the processor between modes infinitely often. In the presence of a transition overhead of t_{tr} time units, we must introduce a non-zero resource demand granularity, which we denote by W_{unit}. In other words, the y-axis of shaping curves of the considered shapers are to be quantized in W_{unit} time units.

During the transition time, the processor consumes P_{act} power but does not execute any job. We can model this by considering these transition times as part of the resource demand of the jobs. Let α_{oh} denote the overhead-aware arrival curve of the task. Since, the minimum amount of time the processor remains in the active mode is W_{unit}, the maximum number of transitions between modes of processors in any interval of length Δ is $\lceil \alpha_{oh}(\Delta)/W_{unit} \rceil$. Thus, α_{oh} is related to α by an overhead term given as

$$\alpha_{oh} \geq \alpha + \left\lceil \frac{\alpha_{oh}}{W_{unit}} \right\rceil \cdot t_{tr} \tag{12}$$

Enforcing α_{oh} to be a valid upper arrival curve (should be sub-additive [9]) while satisfying (12), we have

$$\alpha_{oh} = \left\lceil \frac{\alpha}{W_{unit} - t_{tr}} \right\rceil \cdot W_{unit} \tag{13}$$

Let σ_{ch} be the shaping curve defined as before, but with the modified overhead-aware α_{oh} curve

$$\sigma_{ch} = ConvexHull\left(\alpha_{oh}(\Delta - D)\right) \tag{14}$$

Let σ_{ch} be realized by a cascade of leaky buckets with the ith leaky bucket characterized by bucket size b_i and fill rate r_i. We impose the granularity W_{unit} by modifying σ_{ch} such that the ith leaky bucket has bucket capacity $b_i + W_{unit}$ and (10) is modified with the floor operator taken with respect to W_{unit}. We then have a modified shaping curve σ_{gr}

$$\sigma_{gr}(\Delta) = \min_{\forall i} \left\{ \left\lfloor \frac{b_i + W_{unit} + r_i \Delta}{W_{unit}} \right\rfloor \cdot W_{unit} \right\} \tag{15}$$

We discuss again the schedulability and peak temperature analyses with this σ_{gr}. From Theorem 2, shaping with σ_{ch} satisfies the schedulability constraint with the additional transition overhead. From (15), we have $\sigma_{gr} \geq \sigma_{ch}$, and thus $Del(\alpha, \sigma_{gr}) \leq Del(\alpha, \sigma_{ch})$. As a result, shaping with σ_{gr} also satisfies the schedulability constraint. However, since $\sigma_{gr} \geq \sigma_{ch}$, the peak temperature of the processor when shaping with σ_{gr} can be higher than that with σ_{ch}. We can upper-bound the difference in peak temperatures by noting that $\sigma_{gr} \leq \sigma_{ch} + W_{unit}$. Therefore, the peak temperature under the shaping curve σ_{gr} is less than the temperature of the processor on running for an additional W_{unit} amount of time in the active mode, after reaching the peak temperature under shaping curve σ_{ch}. For very large values of W_{unit} this difference would increase, and it can lead to large peak temperatures. On the other hand, too small a value of W_{unit} would result in a large overhead-aware arrival curve, α_{oh}, as the processor is switching modes more often. This can lead to larger peak temperatures at the best, and non-schedulability at the worst. Thus, setting the value of W_{unit} must be carefully done, as we illustrate in the Section 8.1.

6. MULTIPLE TASKS SCHEDULED BY AN EDF SCHEDULER

So far, we have considered a single stream of jobs. However, in practice several tasks with different arrival curves and deadlines are co-scheduled. We consider a system where the Earliest-Deadline First (EDF) policy is used to arbitrate between several such tasks. The EDF queue is populated first and then the shaper works on this queue (Figure 2).

Let τ_i denote the ith task with arrival curve α_i and relative deadline D_i. For this task-set we can obtain the demand bound function, **dbf**, defined as

$$\mathbf{dbf} = \sum_{\forall\ i} \alpha_i(\Delta - D_i). \tag{16}$$

We define σ_{ch}, with the **dbf** instead of $\alpha(\Delta - D)$ as

$$\sigma_{ch} = ConvexHull(\mathbf{dbf}) \tag{17}$$

In essence, EDF arbitration allows us to coalesce the different tasks into an equivalent task that is characterized by the

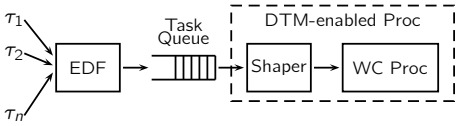

Figure 2: Block diagram for multiple input streams

```
time = 0
prevTime = 0
timer = 0
f_i = b_i + W_unit,   ∀ i

disableInterrupt
runShaper
```

(a) `init`

```
startTime = time
f_i = f_i − W_unit   ∀ i
while((time ≤ startTime + W_unit)
  and (!taskQueueEmpty))
    execute task from task queue
end while
runShaper
```

(b) `runTask`

```
f_i = min{b_i + W_unit, ...
  f_i + (time − prevTime) · r_i), b_i + W_unit}
prevTime = time
if f_i ≥ W_unit,   ∀ i then
   if !task_queue_empty then
      runTask
   else
      enableInterrupt
      force processor into idle mode
   end if
else
   t_sleep = max_∀i{(W_unit − f_i)/r_i}
   timer = t_sleep
   enableInterrupt
   put processor in idle mode
end if
```

(c) `runShaper`

```
Raised if (interruptEnabled
  and !taskQueueEmpty
  and timer == 0)

disableInterrupt
timer = 0
put processor in active mode
runTask
```

(d) `ISR`

Figure 3: Implementation of a leaky bucket shaper

dbf. Thus, with σ_{ch} defined as above, the results presented in Section 4 and 5 apply for such task-sets as well.

7. IMPLEMENTING GREEDY SHAPERS

In this section, we discuss an implementation of the leaky bucket shaper. Our implementation requires three hardware features: (a) an incrementing time variable *time*, (b) a timer that can be programmed to a given positive value and which counts down to 0 and remains there until programmed again, and (c) an interrupt signal which is raised based on the states of the EDF task-queue and the timer. These are standard features supported by most systems. Importantly, our implementation does not require any temperature sensors.

The implementation is described as different routines in Figure 3. The `init` routine is used to initialize the system. The `runTask` routine is called when the processor is to execute tasks. After running tasks for W_{unit} amount of time or until the task-queue is empty, whichever is earlier, the control is passed on to `runShaper` routine. The `runShaper` routine implements the shaping of the tasks based on the leaky bucket parameters derived in (15). If according to the shaping curve, the execution of jobs has to be delayed, then the timer is programmed accordingly. `ISR` is the interrupt service routine that would then put the processor in the active mode at the programmed time if there are jobs to execute. Apart from these routines, the EDF scheduler runs in the background and appropriately populates the task-queue.

The total overhead of `runShaper` is within tens of instructions, a negligible overhead for most application domains. Furthermore, a key advantage of the implementation is that during run-time no additional information about the jobs or the state of the system is required. For instance, we do not require any information about the currently executing job: what its execution time is, how much of that execution has already completed, or if it completes before its stipulated worst-case execution time (WCET). Indeed, at design time such information has been used to characterize the arrival curves and derive the shaping curves. But after the analysis, all the information gained is fully represented in the bucket sizes and fill rates of the leaky buckets. The shaping is entirely based on availability of some job in the task-queue and on the working of the processor, which is suitably and minimally logged in the fill values of the leaky buckets.

ρ_{idl}	ω_{idl}	ρ_{act}	ω_{act}	t_{tr}	G	C	T_{amb}
0.1 W/K	-25 W	0.1W/K	-11 W	0.1ms	0.3 W/K	0.03 J/K	300 K

Table 1: Considered power and thermal parameters

	Video	Audio	Network
Period	0.2	0.2	0.1
Jitter	0.05	0.05	0.03
WCET	0.06	0.03	0.02
Deadline	0.2	0.2	0.1

Table 2: Parameters of tasks in the video conferencing application. All times are in seconds

8. EXPERIMENTAL RESULTS

We consider an ARM-based processor, the power parameters for which are obtained from [12]. The thermal parameters are typical parameters as obtained from [8]. All parameters are summarized in Table 1.

8.1 Computing the optimal shaper

Consider a video-conferencing application that consists of three tasks: a video codec, an audio codec and a network process. The parameters of the tasks are shown in Table 2. We first compute the peak temperature of the system for a workload conserving processor serving the three tasks. From the analytical approach presented in [1], we obtain the peak temperature as $T_{wc}^{max} = 346$ K. We confirm the validity of this bound by generating 20 random traces of jobs and plotting the evolution of temperature for these traces in Figure 4(b). The light points indicate evolution of temperature for all traces, while the dark line indicates the evolution of temperature for an example trace.

We now design the shaping curve by finding the optimal W_{unit}. For different values of W_{unit} in the range $[0.2, 2]$s, we obtain the corresponding overhead-aware **dbf** functions and then compute the leaky bucket shapers as discussed in Sections 4 and 5. For each case, we obtain the peak temperature and plot it in Figure 4(a). As discussed, for too small and too large W_{unit}, the peak temperature is larger. The optimal solution is at $W_{unit} = 0.51$s with a maximum temperature $T_{sh}^{max} = 338$K. We again confirm this bound by random simulations shown in Figure 4(c). Thus, with the obtained shaping curve, we improve the peak temperature by 8K in comparison to the workload conserving case.

The advantage of shaping in terms of lower peak temperatures can be translated to improved workload supported for a given peak temperature. Consider for instance that to support higher video resolution the WCET of the video codec is increased. We would like to identify the largest C_1

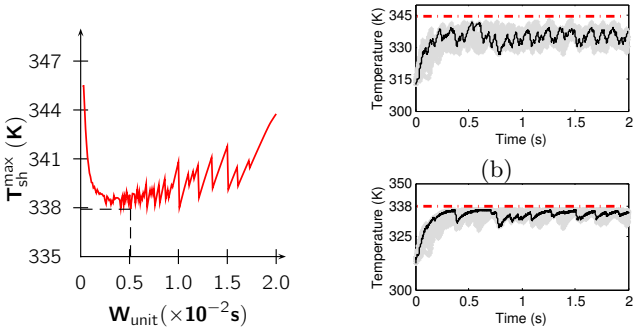

Figure 4: Searching for the optimal shaper

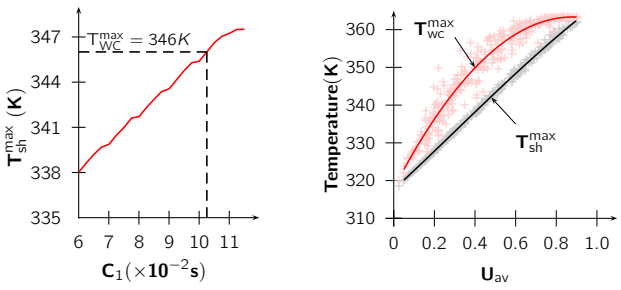

Figure 5: Quantitative advantage of shaping

that can be supported, with shaping, such that the peak temperature is smaller than the peak temperature with no shaping $T^{max}_{wc} = 346$K. We vary C_1 in the range $[0.06, 0.12)$s, and for each value obtain the optimal peak temperature with shaping. From the plot in Figure 5(a), we infer that with shaping of tasks, the processor is able to execute the three tasks with $C_1 = 0.102$s with the peak temperature $T^{max}_{sh} \leq 346$K. Thus, for the same peak temperature obtained with $C_1 = 0.06$s for the workload conserving execution, by shaping the tasks the processor can support $C_1 = 0.1$s: a substantial **66.67%** greater WCET.

8.2 Random task sets

To quantify the advantage obtained with shaping, independent of the task set, we perform an experiment with random task sets. We consider two periodic tasks with randomly generated periods p_1 and p_2 with mean values of 0.2s each, and jitters $p_1/2$ and $p_2/2$, respectively. The WCETs of tasks are also randomly generated with mean as $p_1/4$ and $p_2/4$. For each case, we obtain the highest temperatures for the workload conserving processor (T^{max}_{wc}) and for shaping with the optimal leaky bucket shaper (T^{max}_{sh}). In Figure 5(b), we plot T^{max}_{wc} and T^{max}_{sh} against the average utilization of the task-set defined as $U_{av} = C_1/p_1 + C_2/p_2$, for 500 test cases. Light stars represent results of individual cases, while the dark lines represent the mean values. We denote the mean values as $\bar{T}^{max}_{wc}$ and $\bar{T}^{max}_{sh}$.

Both $\bar{T}^{max}_{wc}$ and $\bar{T}^{max}_{sh}$ are increasing functions of the utilization U_{av}. In all cases, $\bar{T}^{max}_{sh} < \bar{T}^{max}_{wc}$, with a significant average difference of **8.8K**. We can alternately interpret this improvement in performance terms for a given maximum allowed temperature $\bar{T}^{max}$. For the considered system, if the cooling system is efficient enough to allow for a very high temperature $T^{max} > 360$K, then shaping tasks is not important as workload conserving execution itself supports a large utilization. For unrealistically small temperature bounds $T^{max} < 330$K, again, shaping does not provide much benefit as both methods support only small utilizations. However, for realistic values of T^{max} between 330 K to 360 K, shaping tasks does indeed lead to large improvements in the utilization of the system. From our data, on average, shaping tasks substantially increases the supported utilization: by over **40%** in comparison to the workload conserving execution. Further, with the shaping of tasks, an increase in utilization of the system comes at the cost of almost linear increase in the highest temperature. In contrast $\bar{T}^{max}_{wc}$ is concave, reaching high temperatures for lower utilizations.

9. CONCLUSIONS

To face the challenging problem of thermal management in hard real-time systems, we proposed a DTM technique to appropriately delay execution of jobs by using shapers. For the class of leaky buckets we proved that the convex-hull-based shaping curve optimally reduces the peak temperature of the system, while ensuring that each job completes within its deadline. The advantage of our approach is that it can be applied even when the trace of job arrivals is not known, but instead a more general arrival curve model is known. When non-zero overhead are incurred in transitioning to the idle mode, we presented how to modify the optimal shaper by searching for an optimized shaping granularity. We extended our results to consider multiple input streams arbitrated using the EDF policy. We presented a lightweight implementation of the shaper with minimal run-time requirements. Experimental data show that shaping input tasks leads to large improvements (8K on average) in the peak temperature, or equivalently, large improvements (40% on average) in the supported utilization for a given peak temperature.

Acknowledgments

This work is supported by PRO3D project financed by the European Community FP7 programme (FP7-ICT-248776).

10. REFERENCES

[1] D. Rai, *et. al.* Worst-Case Temperature Analysis for Real-Time Systems. In *DATE*, 2011.

[2] A. Kumar, *et. al.* Hybdtm: a coordinated hardware-software approach for dynamic thermal management. In *DAC*, 2006.

[3] N. Bansal, *et. al.* Dynamic speed scaling to manage energy and temperature. In *FOCS*, 2004.

[4] D. Brooks and M. Martonosi. Dynamic thermal management for high-performance microprocessors. In *HPCA*, 2001.

[5] J.-J. Chen, *et. al.* Proactive speed scheduling for real-time tasks under thermal constraints. In *RTAS*, 2009.

[6] L. Georgiadis, *et. al.* Efficient network QoS provisioning based on per node traffic shaping. *IEEE/ACM Trans. Netw.*, 4(4), 1996.

[7] Y. Liu, *et. al.* Accurate temperature-dependent integrated circuit leakage power estimation is easy. In *DATE*, 2007.

[8] K. Skadron, *et. al.* Temperature-aware microarchitecture. In *ISCA*, 2003.

[9] L. Thiele, *et. al.* Real-time calculus for scheduling hard real-time systems. In *ISCAS*, 2002.

[10] E. Wandeler, *et. al.* Performance analysis of greedy shapers in real-time systems. In *DATE*, 2006.

[11] S. Wang and R. Bettati. Reactive speed control in temperature-constrained real-time systems. *Real-Time Systems*, 39(1-3), 2008.

[12] P. Kumar and L. Thiele Thermally Optimal Stop-Go Scheduling of Task Graphs with Real-Time Constraints In *ASPDAC*, 2011.

[13] S. Zhang and K. S. Chatha. Approximation algorithm for the temperature-aware scheduling problem. In *ICCAD*, 2007.