Reliability-aware Thermal Management for Hard Real-time Applications on Multi-core Processors

Vinay Hanumaiah
Electrical Engineering Department
Arizona State University, Tempe, USA
Email: vinayh@asu.edu

Sarma Vrudhula
Computer Science Engineering Department
Arizona State University, Tempe, USA
Email: vrudhula@asu.edu

***Abstract*—Advances in chip-multiprocessor processing capabilities have led to an increased power consumption and temperature hotspots. Reducing the on-die peak temperature is important from the power reduction and reliability considerations. However, the presence of task deadlines constrain the reduction of peak temperature and thus complicates the determination of optimal speeds for minimizing the peak temperature. We formulate the determination of optimal speeds for minimizing the peak temperature of execution with task deadlines as a quasiconvex optimization problem. This formulation includes accurate power and thermal models with the leakage power dependency on temperature. Experiments demonstrate that our approach is very flexible in adapting to various scenarios of workload and deadline specifications. We obtained an 8 °C reduction in peak temperature for a sample execution of benchmarks.**

I. Introduction

To boost performance without a drastic increase in power consumption, the microprocessor industry has adopted the multicore strategy. The idea is to increase performance by parallelizing computation over multiple cores, while operating each core at a lower power dissipation. The lower power dissipation of each core is achieved by reducing the complexity and functionality of each core, and/or operating each core at a lower frequency and voltage. With continued miniaturization, industry projects processors with many hundreds of cores in the near future. The multicore strategy is even being applied to mobile and handheld battery powered devices.

With hundreds or even tens of cores, it will neither be practically feasible nor economical to design a package that will dissipate the maximum possible heat generated by all the cores running at the maximum frequency. Realistically, the package will be designed to dissipate close to the average power dissipation. This has led to investigation of *dynamic thermal management* (DTM) techniques, such as dynamic voltage and frequency scaling (DVFS) [1], [2] and dynamic thread migration [1], [3]. Since DTM techniques will be used much more often in multicore than in single core processors, optimal DTM techniques are critically important to reduce performance degradation.

This work was supported in part by NSF grant CSR-EHS 0509540, NSF-IUCRC on embedded systems, and by a grant from Science Foundation Arizona (SFAz) and Stardust Foundation.

978-3-9810801-7-9/DATE11/©2011 EDAA

The recent work on DTM for multicores has focused on optimizing performance (throughput or makespan) [4], [5], design of closed loop DTM controllers [2] and design space exploration [6] in the presence of thermal constraints. However a more critical issue with high performance multicores will be *reliability* due to increasing die temperatures. A 10-15 °C increase in temperature can reduce the lifespan of a device by half [7]. The ITRS has predicted a rapid onset of significant lifetime reliability problems. It is expected in the future, processor cost and performance specifications will be significantly affected by the lifetime reliability and will take over as the primary factor in the processor design, superseding performance requirements.

Several researchers have recently focused on DTM targeting reliability. References [8], [9] proposed models for reliability estimation at the chip-level and showed that the reliability can be traded with performance. The authors of [10] provided an analysis on the trade-off of power consumption with performance and reliability through the use of various power management techniques. In [1], a mixed-integer linear program was used to determine the optimal task schedule to reduce the peak temperature with task deadlines. A heuristic was developed in [11] to sequence the tasks on a single-core processor to minimize the peak temperature with timing constraints. The authors of [12] evaluated a large number of techniques to study the effect of job scheduling and power management techniques on the system reliability.

The above works attempted to address the issue of system reliability from the DTM perspective and proposed techniques for optimal task scheduling and sequencing. However, these previous works do not address the important transient optimal control of frequencies of the processor. The determination of these frequencies are complicated by the non-linear cyclic relation between the leakage power and the temperature. This determination of frequencies gets even more complicated with the addition of task deadlines. In this paper, we present the first quasiconvex programming solution for determining the speeds of a multi-core processor to ensure that all tasks meet their deadlines with minimum peak temperature. Experimental results demonstrate that our approach helps in improving the lifetime of devices by reducing the peak temperature by 8 °C for a sample execution of tasks from SPEC benchmarks. We also show a practical implementation of our approach with

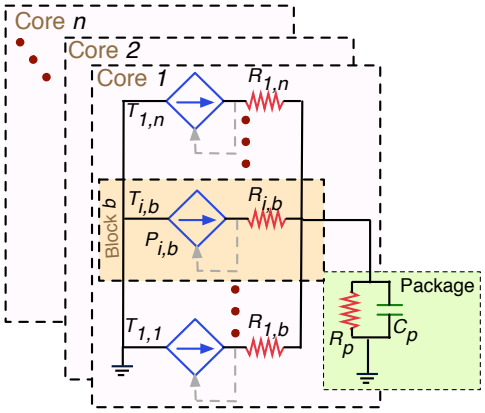

Fig. 1. Simplified high-level multi-core thermal model.

discrete speed states.

II. Thermal and Reliability Models

Consider an n core processor with each core executing a single task (no SMT). Task i begins at time $t_{s,i}$ and should finish its execution before deadline $t_{d,i}$. Without loss of generality $t_{s,i} \leq t_{s,i+1}$ is assumed. The speed s_i of core i is continuous and normalized over [0,1]. Every task is assumed to run for atleast the die-thermal time constant (few ms). Majority of scientific workload falls under this case.

Reliability of a core is given by the reliability of the weakest interconnect in the core. The granularity of our thermal and reliability analysis is at the level of functional blocks of cores. Every core has a block which remains the hottest irrespective of the frequency of operation, called the *hottest block*. Hence the reliability of a core can be computed by calculating the reliability of the hottest block alone. Electromigration (EM) plays a major role in the interconnect breakdown, thereby reducing the lifetime of a processor. The mean time to failure (MTTF) $t_{f,i}$ of core i caused due to EM is given by Black's equation [8].

$$t_{f,i} = \frac{A}{j_{i,h}^n} e^{\frac{Q}{kT_{i,h}}}, \tag{1}$$

where A is a constant based on interconnect structure, Q is the activation energy (0.6 ev for Aluminum), $j_{i,h}$ and $T_{i,h}$ are the current density and the temperature of the hottest functional block h in core i. k is Boltzmann's constant. $n = 1$ or $n = 2$ depending on the failure mode. The only issue of relevance of (1) is that the MTTF decreases exponentially with increase in the temperature. Thus to maximize the lifetime of a device, the peak temperature of operation has to be minimized.

HotSpot [13] is used to model the thermal behavior of processors. HotSpot uses circuit-thermal analogy to model the heat spreading and storage behavior of processors. Each core is divided into m thermal block in the die and the thermal interface material (TIM) according to their functionality, followed by spreader and heat sink layers in the package.

It has been observed that the heat conducting ability of lateral resistances between the thermal blocks are approximately four times lower than the vertical resistances in the die and the TIM layers [5]. Based on this observation, the lateral resistances can be ignored to simplify the analysis without significant loss in accuracy. Also since our tasks run longer than the die thermal time constants, the die capacitances can be neglected. This simplified high-level thermal model is described in Fig. 1 with $P_{i,b}$, $R_{i,b}$ and $T_{i,b}$ representing the power source, the vertical resistance and the temperature of block b in core i. R_p and C_p represent the lumped package resistance and capacitance respectively.

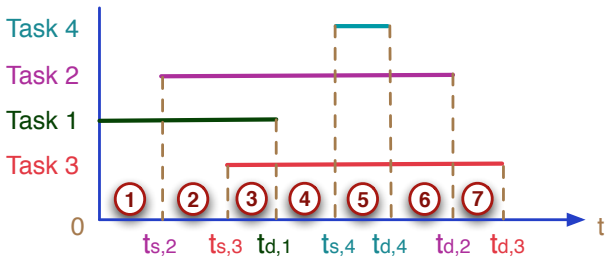

Fig. 2. Creation of slots based on start and end times of tasks.

The important power and temperature relations of the simplified thermal model are described here [5]. The temperature $T_{i,b}$ depends linearly on its dynamic power $P_{i,b,d}$ and exponentially on its leakage power $P_{i,b,l}$. In order to keep the analysis tractable, the exponential relationship has to be linearized. This is achieved with the use of piece-wise linearization. Let $\zeta_{i,b} \triangleq (1 - k_{i,b}R_{i,b})^{-1}$ be the leakage coefficient, where $k_{i,b}$ is the slope of the leakage power vs the temperature of block b in core i. It can be shown that

$$T_{i,b}(t) = \zeta_{i,b}[T_p(t) + (P_{l,i,b} + s_i(t)P_{d,i,b})R_{i,b}], \tag{2}$$

where T_p is the lumped package temperature. The corresponding power consumption of the block is given by

$$P_{i,b}(t) = \zeta_{i,b}(P_{l,i,b} + s_i(t)P_{d,i,b}) + (\zeta_{i,b} - 1)T_p(t)/R_{i,b}. \tag{3}$$

The package temperature T_p is computed by solving the following ordinary differential equation.

$$\frac{dT_p(t)}{dt} = -\frac{T_p(t)}{\tau_p} + \frac{P_T(t)}{C_p}. \tag{4}$$

$P_T(t) = \sum_{i=0}^{n}\sum_{b=0}^{m} P_{i,b}(t)$ is the total power consumption of all blocks in all cores. τ_p is the package thermal time constant.

III. Problem Statement and Approach

A. Problem Description

Consider an n core processor with each core i executing a task starting at time $t_{s,i}$ with N_i instructions to be executed within deadline $t_{d,i}$. Fig. 2 shows an example of a four-core processor executing four tasks starting and ending at various times. The slots are numbered in an increasing fashion. Given this, the problem is to determine the optimal transient speeds of cores such that the deadline constraints for the tasks are

satisfied while minimizing the peak temperature of operation T_{max}. The problem is formulated as

$$\min_{\mathbf{s}(t), T_{max}} \quad T_{max}, \tag{5}$$

$$s.t. \quad \frac{d\mathbf{n}(t)}{dt} = \mathbf{s}(t), \ \forall t, \tag{6}$$

$$\mathbf{n}(\mathbf{t}_s) = 0, \quad \mathbf{n}(\mathbf{t}_d) = \mathbf{N}, \tag{7}$$

$$T_{i,b}(t) = \zeta_{i,b}[T_p(t) + (P_{l,i,b} + s_i(t)P_{d,i,b})R_{i,b}], \ \forall t, \tag{8}$$

$$\mathbf{T}(t) \leq T_{max}, \ \forall t, \quad \mathbf{T}(0) = \mathbf{T}_0. \tag{9}$$

In the above formulation, the objective is to minimize the peak temperature of execution while satisfying the deadline constraints specified by (7), where $n_i(t)$ is the number of instructions completed by task i in time t. The relationship between the instructions completed $\mathbf{n}$ and the core speeds $\mathbf{s}$ is given by (6). $\mathbf{t}_s$ and $\mathbf{t}_d$ are the vector notations of the start and the end times respectively. The core speeds $\mathbf{s}$ and the peak temperature T_{max} are the variables of optimization. Note that the cores i vary from 1 to n and the blocks b from 1 to m. Hence the dimensions of $\mathbf{n}$ and $\mathbf{s}$ are $n \times 1$, while the dimension of $\mathbf{T}$ is $nm \times 1$.

The above optimization is a very challenging problem in optimal control theory due to the presence of the boundary conditions and the non-linear constraints. However, the optimal *policy* can be determined analytically, whereas the parameters of the analytical solution must determined numerically. In the following section we present an outline of how the solution is constructed.

B. Solution Outline

The entire duration of execution is partitioned into several time slots based on the task deadlines (see Fig. 2), and the above optimization problem is solved for each time slot in three steps.

1) Determine the optimal speed profiles of the multi-core processor to maximize the throughput (sum of speeds) for given tasks under a fixed maximum temperature and no deadline constraints.
2) Solve the unknown parameters of the above speed profiles to satisfy the boundary conditions imposed by the start and the end times of tasks.
3) Find the minimum peak temperature under which Step 2 is still satisfied.

The above steps lead to an optimal solution for the following reasons. The first step yields a parametric solution which represents the set of all possible optimal solutions over the space of all its parameters. The second step finds the subset of the solutions (characterized by a set of parameters) that satisfy the boundary conditions. One of those parameters is T_{max}. Among the solutions found in Step 2, the third step finds the solution that corresponds to the minimum T_{max}.

It will be shown later in Section IV-B that the solution of Step 2 requires the knowledge of the initial package temperature of the slot in consideration and either the final temperature of the slot or the number of instructions to be completed within the slot. Since the final package temperature of a slot is also the initial package temperature of the next slot, knowing one of them is sufficient. It will also be shown later in Section IV-C that Step 2 is a quasiconvex function of the initial package temperature of the slot and the peak temperature T_{max}. By quasiconvex, we mean that the function is unimodal and a unique minimum value. Thus a quasiconvex optimization problem can be solved to determine the minimum T_{max}. Each of these steps will be explained in detail in the following sections.

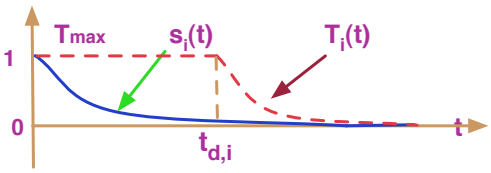

Fig. 3. Typical speed and temperature plots for the optimal makespan minimization.

IV. Optimal Solution

A. Optimal Speed Profile for Tasks with no Deadlines and Constant Maximum Temperature

Here we summarize the results for speed control of tasks in a multi-core processor to minimize the latest task completion time (which also maximizes the overall throughput) under a constant maximum temperature constraint [4]. Note that we are neglecting all the tasks deadlines, and the maximum speed is constrained by the maximum temperature alone.

The optimal speed of core i for minimizing the makespan is the one which maintains the temperature of the hottest block at the maximum temperature. It can be shown that such a speed will have the following parametric exponential form.

$$s_i(t) = s_{i,0}e^{-\frac{t}{\tau_p}} + s_{i,ss}\left(1 - e^{-\frac{t}{\tau_p}}\right), \tag{10}$$

where $s_{i,0}$ is the initial speed of core i determined by setting the temperature of its hottest block (h) $T_{i,h} = T_{max}$ in (2) and solving for speed s_i. $s_{i,ss}$ is the final steady state temperature which can be computed by equating the R.H.S of (4) to zero.

This speed curve is shown in Fig. 3. We refer to this policy as the *max-throughput* policy.

B. Optimal Speed Profile Satisfying Task Deadlines under Constant Maximum Temperature

If the speed function in (10) satisfies the task deadlines, then it is the optimal solution for the problem with deadlines. In general, the speed function of the *max-throughput* solution need not be optimal in the presence of deadlines. This is demonstrated with the aid of Fig. 4. Consider a sample execution of the four tasks, where task 2 has its deadline at end of the slot and dissipates higher power than task 1. Fig. 4(a) shows the max-throughput execution of four tasks within a slot. Task 1 executes at a higher frequency than task 2 due to its lower power dissipation and thus maximizing the throughput while satisfying the constraints on the peak

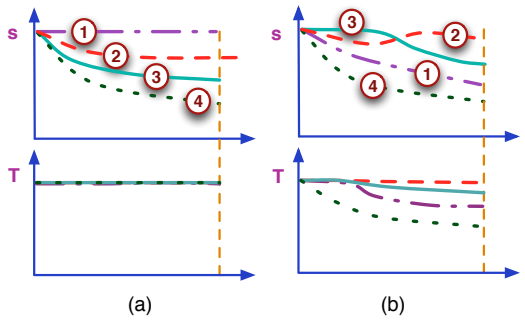

Fig. 4. Comparison of (a) max-throughput policy with (b) the optimal policy in meeting the tasks deadlines within a slot.

temperature. However, due to slower execution of task 2, task 2 deadline is violated. This can be avoided by selectively executing task 2 faster as shown in Fig. 4(b). The problem is to determine the core speeds optimally such that the deadlines are satisfied. Note that this modification in speed of task 2 affects the speeds of other tasks. We call the task with nearest deadline as the *critical task* (task 2) and the rest of the tasks as non-critical tasks.

1) Speed of the critical task: Since the critical task i requires the maximum execution speed, its speed function is determined by the *max-throughput* policy as given by (10). In (10), the initial speed $s_{i,0}$ is fixed by the initial package temperature. Hence we only need to determine the steady-state speed $s_{i,ss}$ such that the task i meets its deadline.

Let l be the current slot, which extends from time t_{l-1} to t_l. Let N'_i be the remaining number of instructions of the critical task i to be completed in the duration of the slot. Integrating s_i over the slot duration gives N'_i, i.e $N'_i = N_i - \int_0^{t_{l-1}} s_i(t)dt$, where N_i is the original number of instructions to be completed. Substituting s_i from (10),

$$(s_{i,ss} - s_{i,0})\tau_i \left(e^{-\frac{t_{l,i}}{\tau_i}} - e^{-\frac{t_{l-1,i}}{\tau_i}} \right) + s_{i,ss}(t_{l,i} - t_{l,i-1}) = N'_i. \quad (11)$$

The above equation is solved for $s_{i,ss}$ and substituted in (10) to obtain the critical task speed. The non-critical tasks speeds are determined from the remaining total power budget (total power budget - critical task power). Note that the total power budget is fixed due to the bounds on the maximum temperature.

2) Speeds of non-critical tasks: Since the critical core runs at the maximum thermally feasible speed, the temperature of its hottest block will be at T_{max}. Substituting $T_{i,h} = T_{max}$ in (2), where h denotes the hottest block, gives the corresponding package temperature for every instant of time.

$$T_p(t) = [T_{max}/\zeta_{i,h} - (P_{l,i,h} + s_i(t)P_{d,i,h})R_{i,h}], \ \forall t. \quad (12)$$

The differential equation (4) must now be solved with T_p given by (12). This is solved numerically by partitioning the current slot l into small scheduling intervals of length t_{sched}. The length of these scheduling intervals are of the order of the die thermal time constant (few ms). Note that the speed is assumed to be constant in these small intervals. Taking the difference of the package temperature between adjacent time intervals in (12) gives an approximation to $\frac{dT_p(t)}{dt}$, which can then be substituted in (4) to get the total power consumption $P_T(t_{sched})$ over that time interval.

Let $P'_T(t_{sched}) = P_T(t_{sched}) - P_i(t_{sched})$ be the remaining total power budget, where P_i is the power consumption of the critical task i determined from (3). This remaining power budget needs to be allocated among the remaining tasks $i+1$ to n in such a way that the tasks with earlier deadlines are allocated the maximum share. To allocate this power budget, we first determine the maximum thermally feasible speed for task $i+1$ - the next critical task. This is obtained by setting $T_{i+1,h} = T_{max}$ and solving for s_{i+1} in (2). This will dissipate P_{i+1} amount of power which is computed from (3). This power is now subtracted from $P'_T(t_{sched})$ and the remaining total power is allocated to the remaining tasks by repeating the above steps for task $i+2$ and so on. This procedure is continued until all the power budget has been allocated. It is possible that a task may not be allocated any power, in which case that task does not execute in that interval.

The above procedure is applicable even in the case of multiple tasks with same deadline. In this case, the speed function curve is derived for only one of the critical tasks and the resultant total power is distributed among the remaining tasks including the other critical tasks, as explained before for non-critical tasks. The overall procedure of determining the feasible speeds satisfying both thermal and deadline constraints within a slot is summarized in Procedure `find_spd_slot`.

Input: T_{p0}, T_{max}, $\mathbf{t_d}$ or T_{pf} (final pkg. temp)
Output: Core speeds $\mathbf{s}$
begin
  Run *max-throughput* policy;
  if *deadline constraint for critical task i is met* **then**
    Accept $\mathbf{s}$ returned by the policy;
  else
    Determine s_i with constraint $t_{d,i}$ from (11);
    Obtain $T_p(t)$ from (12);
    for *every* t_{sched} **do**
      Find approx. $\frac{dT_p(t)}{dt}$ over t_{sched};
      Substitute $\frac{dT_p(t)}{dt}$ in (4) to get $P_T(t_{sched})$;
      Compute $P_i(t_{sched})$ from (3);
      $P'_T(t_{sched}) = P_T(t_{sched}) - P_i(t_{sched})$;
      for *task c from $i+1$ to n* **do**
        Find $P_c(t_{sched})$ with $T_{c,h} = T_{max}$ in (3);
        $P'_T(t_{sched}) = P'_T(t_{sched}) - P_c(t_{sched})$;
        if $P'_T(t_{sched}) \leq 0$ **then**
          $s_c, \ldots, s_n = 0$; break;
        end
      end
    end
  end
end

Procedure `find_spd_slot`

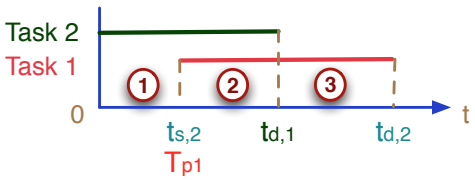

Fig. 5. Two task example to demonstrate the quasiconvexity of (14).

C. Quasiconvex Programming Solution for Minimizing Peak Temperature

From the previous section we know how to determine the optimal core speeds to satisfy task deadlines within a slot. We now need to determine the global minimum peak temperature such that all tasks satisfy their deadlines. We note that in order to determine the minimum peak temperature, the initial package temperature of each slot has to be determined optimally. This is necessary as the optimal core speeds in each slot are guided by the initial package temperatures. We formulate the problem of determining the optimal initial package temperatures of slots and the minimum peak temperature as a quasiconvex optimization problem.

$$\max_{\mathbf{T_p}, T_{max}} \quad T_{max}, \tag{13}$$

$$s.t. \quad \mathbf{s} = \texttt{find\_spd\_slot}(T_{p,l}, T_{max}),\ \forall l, \tag{14}$$

$$\mathbf{T}_p \leq T_{max}, \quad T_p(0) = T_{p0}. \tag{15}$$

Here $T_{p,l}$ is the initial package temperature of slot l. *In the interest of clarity and lack of space, the proof of the quasiconvexity of the above formulation is omitted.*. It is trivial to show that the objective (13) is quasiconvex. Procedure `find_spd_slot` finds a feasible solution for Formulation (6) – (9) for a slot l. We give an intuitive explanation for the quasiconvexity of Procedure `find_spd_slot` in (14) by considering a two task scenario with start and end times as shown in Fig. 5. In this case, there is only one package temperature T_{p1} to be determined. Let the optimal $T^*_{p1} \in [T_{p1,min}, T_{p1,max}]$. Consider $T_{p1} = T_{p1,max} + \epsilon$, where ϵ is any small value, violates the deadlines in either slot 2 or 3. This can be due to the lower initial speed in slots 2 and 3 caused by higher T_{p1}, which affects the completion times of task 1 and 2. Increasing ϵ to any higher value will only reduce the initial speed in slot 2 and 3 further. Similarly on the other hand, consider $T_{p1} = T_{p1,min} - \epsilon$ leading to deadline violation of task 1 due to lower final speed of task 1 in slot 1. Increasing ϵ to any higher value will only decrease the speed of task 1 in slot 1. Thus there is a single continuous range of satisfiable values for T_{p1}. Hence (14) is quasiconvex over T_{p1}.

V. Experimental Results

We experimentally verify our policy by simulating SPEC benchmarks on a multi-core version of Alpha 21264. HotSpot [13] and PTScalar [14] were used for thermal and power modeling respectively. The dynamic and the leakage power were limited to 230 W and 60 W respectively. The scheduling interval was set at 10 ms.

Scenario	Parameters		bzip	gap	mcf	twolf
1	Instructions (billions)		220	80	296	308
	Start times (s)		0	10	24	33
	End times (s)		60	45	124	130
	Act. end times (s)	Opt. policy	58.7	34.6	122.2	130
		max-tput.	51.4	31.7	111.3	119.3
2	Instructions (billions)		220	80	296	308
	Start times (s)		0	10	24	33
	End times (s)		45	30	104	112
	Act. end times (s)	Opt. policy	45	29.4	101.7	109.8
		max-tput.	51.4	31.7	111.3	119.3

TABLE I
CHARACTERISTICS OF TASKS USED IN THE EXPERIMENTS.

A. Optimal Policy vs Max-throughput Policy

Here we compare our proposed optimal policy with the *max-throughput* policy [4] for two scenarios listed in Table I. The tasks, their start times, deadlines and the actual end times under both the policies are also listed in the table. For the sake of clarity, only four tasks are used in the experiments, although our method works for any number of cores and tasks. Consider the execution of Scenario 1 under both the optimal procedure and the max-throughput method as shown in Figs. 6 and 7 respectively. We find that both the policies satisfy the deadlines, but our optimal policy executes the tasks at a lower temperature of 102 °C, while the max-throughput policy executes the tasks under the nominal temperature of 110 °C until completion. This reduction in the peak temperature can be even lower if the task deadlines are relaxed much further. It is interesting to note that the task gap finishes its execution early at 34.6 s under the optimal policy when its actual deadline is 45 s. This is because, finishing the task gap any later would lead to deadline violation of atleast one of the tasks with future deadlines with 102 °Cas the maximum temperature.

Next we execute tasks from Scenario 2 under both the max-throughput policy and the optimal policy as shown in Fig. 8 and Fig. 7 respectively. Since the max-throughput policy only tries to maximize the overall completion of instructions it results in the same core speeds for a given set of instructions irrespective of any deadlines (for both Scenarios 1 and 2) and hence the same plots. From the figures, we see that the max-throughput policy fails to satisfy the deadlines for tasks bzip, gap and mcf as it is not able to increase the peak execution temperature beyond 110 °C, while our optimal policy executes the tasks optimally at the peak temperature of 118 °C. The method also selectively, but optimally increases the speeds of the critical tasks to ensure that their deadlines are met. This demonstrates the flexibility and also the optimality of our proposed method in adopting to different tasks scenarios.

Fig. 9 shows the practical speed implementation of the optimal scheduling policy for the workload specified in scenario 2 with eight speed states. The optimal speeds are discretized by selecting the highest discrete speed state that is feasible under the temperature constraint at every instant. Note that due to the discretization of speeds, which is an approximation of the optimal solution, the peak temperature increases by 4 °C.

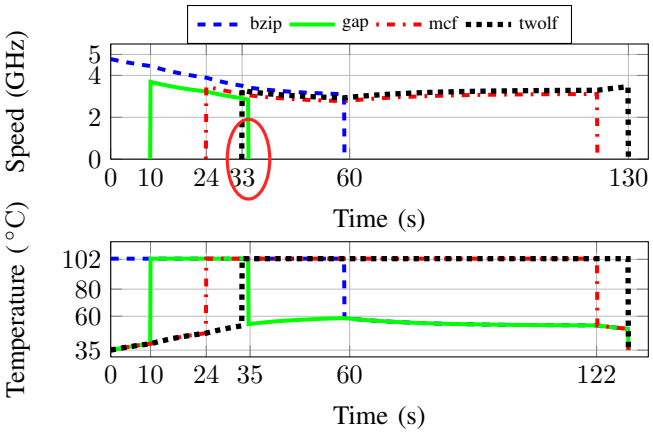

Fig. 6. Optimal speeds and temperatures for Scenario 1.

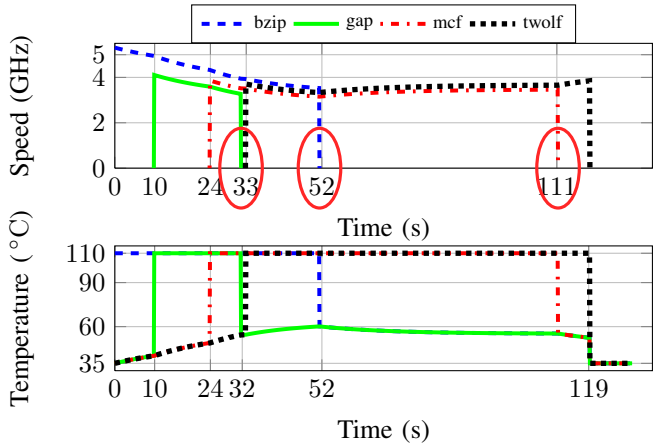

Fig. 7. *Max-throughput* speeds and temperatures for Scenario 1 and 2.

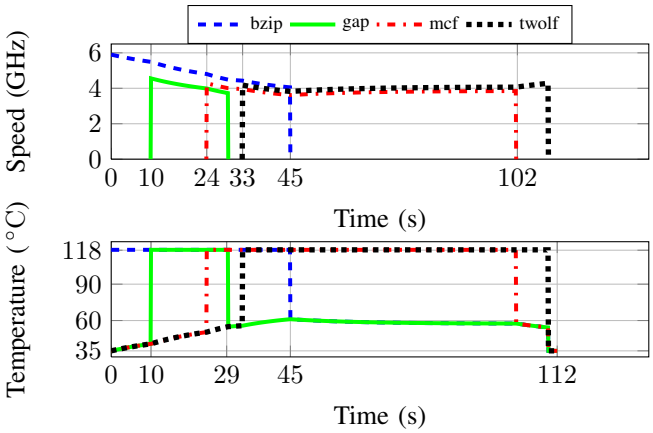

Fig. 8. Optimal speeds and temperatures for Scenario 2.

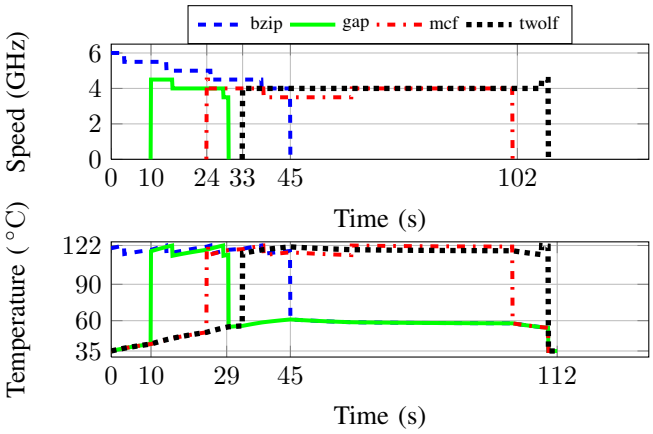

Fig. 9. Discretization of Optimal speeds and temperatures for Scenario 2 with eight speeds.

VI. Conclusion

High operating temperatures combined with device scaling have led to dwindling of lifetime reliability. In this paper, we propose to maximize the lifetime reliability of a many-core processor executing a set of tasks under hard deadline constraints. We formulate the problem as a quasiconvex optimization problem to minimize the peak temperature. Our experimental results demonstrate that our method can adopt to various task scenarios and derive optimal speeds that result in reduced execution temperature. Our work finds application in early phase design space exploration, offline optimal frequency allocation and lifetime reliability optimization.

References

[1] T. Chantem, R. P. Dick, and X. S. Hu, "Temperature-aware Scheduling and Assignment for Hard Real-time Applications on MPSoCs," in *Proc. DATE*, 2008, pp. 288–293.

[2] Y. Wang, K. Ma, and X. Wang, "Temperature-constrained Power Control for chip Multiprocessors with Online Model Estimation," *SIGARCH Comput. Archit. News*, vol. 37, pp. 314–324, 2009.

[3] M. D. Powell, M. Gomaa, and T. N. Vijaykumar, "Heat-and-run: Leveraging SMT and CMP to Manage Power Density through the Operating System," in *Proc. ASPLOS*, 2004, pp. 260–270.

[4] V. Hanumaiah, S. Vrudhula, and K. S. Chatha, "Performance Optimal Speed Control of Multi-Core Processors under Thermal Constraints," in *Proc. DATE*, 2009, pp. 288–293.

[5] R. Rao and S. Vrudhula, "Performance Optimal Processor Throttling under Thermal Constraints," in *Proc. CASES*, 2007, pp. 257–266.

[6] M. Monchiero, R. Canal, and A. González, "Power/performance/thermal Design Space Exploration for Multicore Architectures," *IEEE Trans. Parallel and Distributed Sys.*, 2008.

[7] R. Viswanath *et al.*, "Thermal Performance Challenges from Silicon to Systems," *Intel Technology Journal*, vol. 4, pp. 1–16, 2000.

[8] J. Srinivasan *et al.*, "The Case for Lifetime Reliability-Aware Microprocessors," *SIGARCH Comput. Archit. News*, vol. 32, p. 276, 2004.

[9] E. Karl *et al.*, "Reliability Modeling and Management in Dynamic Microprocessor-based Systems," in *Proc. DAC*, 2006, pp. 1057–1060.

[10] K. Waldschmidt *et al.*, "Reliability-Aware Power Management Of Multi-Core Systems (MPSoCs)," in *Proc. Dynamically Reconfigurable Architectures*, 2006.

[11] R. Jayaseelan and T. Mitra, "Temperature Aware Task Sequencing and Voltage Scaling," in *Proc. ICCAD*, 2008, pp. 618–623.

[12] A. K. Coskun *et al.*, "Evaluating the Impact of Job Scheduling and Power Management on Processor Lifetime for Chip Multiprocessors," in *Proc. SIGMETRICS*, 2009, pp. 169–180.

[13] W. Huang *et al.*, "HotSpot: A Compact Thermal Modeling Method for CMOS VLSI Systems," *IEEE Trans. VLSI Sys.*, vol. 14, pp. 501–513, 2006.

[14] W. Liao, L. He, and K. M. Lepak, "Temperature and Supply Voltage Aware Performance and Power Modeling at Microarchitecture Level," *IEEE Trans. Computer-Aided Design*, vol. 24, pp. 1042–1053, 2005.