Temperature aware Task Sequencing and Voltage Scaling

Ramkumar Jayaseelan Tulika Mitra
Department of Computer Science
National University of Singapore
{ramkumar,tulika}@comp.nus.edu.sg

***Abstract*—On-chip power density and temperature are rising exponentially with decreasing feature sizes. This alarming trend calls for temperature management at every level of system design. In this paper, we propose task sequencing as a powerful and complimentary mechanism to voltage scaling in improving the thermal profile of an embedded system executing a set of periodic heterogenous tasks under timing constraints. We first derive the peak temperature of a repeating task sequence analytically and develop a heuristic to construct the task sequence that minimizes the peak temperature. Experimental evaluation shows that our task sequencing heuristic achieves peak temperature within 0.5^oC of the optimal solution and 7.47^oC lower, on an average, compared to the worst sequence for a large range of embedded task sets. We also propose an iterative algorithm that combines task sequencing with voltage scaling to further lower the peak temperature while satisfying the timing constraints. For embedded task sets, our combined task sequencing and voltage scaling approach achieves, on an average, $2.1^oC - 6.94^oC$ reduction in peak temperature compared to voltage scaling alone.**

I. Introduction

As power density continues to increase exponentially with technology scaling, the resulting rise in on-chip temperature is pushing the limits of packaging and cooling technology [4], [11]. Thus thermal management and innovative cooling solutions have become important aspects of computer system design. The thermal problem is equally prominent in embedded systems, where power densities are expected to rise significantly [22]. Moreover, the constraints on the size and cost of mobile embedded devices do not allow for complex packaging and cooling solutions.

Increased on-chip temperature can result in poor reliability, increased cooling cost and timing errors. The cost of packaging and cooling solutions increases super-linearly with increase in power density [4]. Also the packaging and cooling costs form a significant fraction of the total cost of a computing system [11]. Increased on-chip temperatures can accelerate failure mechanisms, such as dielectric breakdown and electro-migration, resulting in permanent damage to the chip. A $10-15^oC$ reduction in peak temperature can result in 2X increase in the lifetime of the chip [2]. Carrier mobility decreases with increase in temperature; thus higher operating temperatures can result in more frequent transient errors [19]. Leakage current increases exponentially with increasing temperature and the positive feedback between leakage power and temperature can result in a thermal runaway. Hence there is a need to control on-chip temperature at every level of the system design.

Traditionally on-chip temperature has been controlled by employing better packaging and cooling technologies. As on-chip temperature continues to rise, it is prohibitively expensive to employ packaging and cooling solutions for the worst-case temperature [4]. Hence chip packaging is designed for a peak temperature that is less than the worst possible temperature [19]. The operating temperature now needs to be maintained below this maximum specified limit by engaging a host of hardware and software techniques. In the recent years, researchers have explored a variety of schemes for architectural and software level thermal management. Most of these techniques are *reactive or dynamic* in nature. Whenever the on-chip temperature exceeds a predefined threshold, different mechanisms (such as voltage and frequency scaling, fetch throttling [5], scheduling priority adjustments [14], task migrations [10], etc.) can be engaged to reduce the temperature. These dynamic thermal management techniques are more suitable in the context of general purpose systems where there are no constraints on the execution times or the quality of service.

More recently, the thermal aware design problem has been addressed in the context of embedded systems. The key difference is that the application-specific and predictable nature of the embedded systems enable the use of static or design time temperature management techniques. In this paper, we address the problem of *minimizing the peak temperature of a set of periodic heterogenous tasks executing on a processor under timing constraints.* The existing solutions to similar problems [22], [16] employ voltage scaling as the only mechanism for temperature management. We observe that task sequencing has significant impact on the thermal profile and the peak temperature. Therefore, we explore a combination of task sequencing and voltage scaling to optimally reduce the peak temperature.

Why Task Sequencing? In general, tasks exhibit significant variation in terms of their power consumption characteristics [18], [16]. This is because power consumption depends on the circuit activity factor that can vary significantly across tasks [16]. Moreover, modern low-power processors engage aggressive clock gating and static power saving techniques [6], [9], [8] where the idle functional units are either clock gated or completely switched off. The power consumption in such a scenario depends heavily on the usage pattern of the functional units, which again varies significantly across tasks. Variation in power consumption characteristics result in variation of the thermal profile.

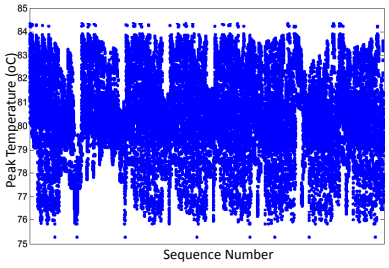

Fig. 1. Peak temperature for all possible task sequences.

We observe that when a task set comprising of such heterogenous tasks executes on a processor, the resulting temperature profile and the peak temperature are highly sensitive to the execution sequence. Figure 1 plots the peak temperatures corresponding to all possible 8! sequences of a task set consisting of eight tasks (crc, epic, gsm, stringsearch, dijkstra, djpeg, adpcm, patricia). A difference of 9.02^oC is observed between the peak temperature of the worst sequence (highest peak temperature) and the best sequence (lower peak temperature).

Our Contributions: Our contributions in this paper are summarized in the following.

- We observe that the peak temperature of a task set comprising of heterogenous tasks is dependent on the execution sequence and analytically derive the peak temperature of a given task sequence.
- Given a task set, we present a heuristic for task sequencing to minimize the peak temperature.
- Given a fixed task sequence and a timing constraint, we design an optimal pseudo-polynomial time algorithm that performs voltage scaling to minimize the peak temperature.
- Given a task set and a timing constraint, we propose an iterative algorithm that combines task sequencing and voltage scaling to reduce the peak temperature. Our iterative algorithm outperforms the optimal voltage scaling algorithm even when the later is fed with the best task sequence (for non-voltage-scaled versions of the tasks).

Organization: The rest of the paper is organized as follows. The background information on the thermal model and the thermal profile of a single task are presented in the next section. Section III presents the formulation of the task sequencing problem and our task sequencing algorithm. Voltage scaling and task sequencing for minimizing the peak temperature are discussed in Section IV. The experimental results are presented in Section V, the related work is presented in Section VI and Section VII concludes the paper.

II. Background

Thermal Model: We choose a lumped RC model proposed by Skadron et al. [19] as our processor thermal model. If the processor dissipates an average power of P Watt over a time interval t, then the temperature $T(t)$ at the end of the time interval is given by

$$T(t) = P \times R + T_{amb} - (P \times R + T_{amb} - T_{init})e^{-t/RC} \quad (1)$$

where R is the thermal resistance measured in $^oC/Watt$, C is the thermal capacitance measured in $Joules/^oC$, T_{amb} is the ambient temperature, and T_{init} is the initial temperature. $T_S = P \times R + T_{amb}$ is the **steady state temperature** associated with an average power dissipation of P Watt.

Thermal Profile of a Task: Let us now look at the thermal profile of an individual task J_i with average power consumption P_i and execution time c_i running on a processor. The thermal model of the processor is given by Eqn 1. Therefore

$$T(c_i) = P_i \times R + T_{amb} - (P_i \times R + T_{amb} - T_{init})e^{-c_i/RC} \quad (2)$$

The steady state temperature of the task J_i is defined as

$$T_{S_i} = P_i \times R + T_{amb} \quad (3)$$

T_{S_i} is the temperature that would be reached if infinite number of instances of task J_i execute continuously on the processor. Let

$$m_i = e^{-c_i/RC} \quad (4)$$

Then substituting into Eqn 2 and rearranging the terms, we get

$$T(c_i) = (1 - m_i)T_{S_i} + m_i T_{init} \quad (5)$$

We observe from Eqn 5 that if $T_{init} < T_{S_i}$, then the temperature rises towards T_{S_i}. Alternatively, if $T_{init} > T_{S_i}$, then the temperature falls towards T_{S_i}.

III. Task Sequencing

In this section, we concentrate on the task sequencing problem. Given a periodic set of heterogenous tasks (i.e., tasks with different thermal profiles), our goal is to construct a task sequence that minimizes the peak temperature. However, a proper formulation of this problem first requires a clear definition of the thermal profile and the peak temperature of a task sequence. So we first proceed to analytically model the thermal profile of a task sequence.

A. Thermal Profile of a Task Sequence

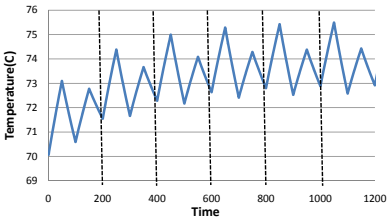

Fig. 2. Thermal profile of a repeating sequence of tasks.

Let us consider a particular sequence $\mathcal{L} = \langle J_1, \ldots, J_N \rangle$ of N tasks with execution times $c_1, \ldots, c_N$, average power $P_1, \ldots, P_N$, and the corresponding steady state temperatures $T_{S_1}, \ldots, T_{S_N}$ where $T_{S_i} = P_i \times R + T_{amb}, 1 \leq i \leq N$. As the task set is periodic, the sequence $\mathcal{L}$ repeats itself infinitely. Figure 2 shows the thermal profile of a repeating sequence of 4 tasks. It is interesting to observe that starting from an initial temperature, the processor temperature rises as the sequence repeats itself. But it gradually reaches a steady state where the thermal profile of the sequence exhibits a recurring pattern. The existence of the this recurring pattern is a result of the fact that (a) given a starting temperature and a repeating sequence, the temperature at the end of each iteration of the sequence is either non-increasing or non-decreasing (this can be proved using induction on the number of tasks and Equation 5), and (b) the final temperature at the end of the sequence is upper (lower) bounded by the steady state temperature of the hottest (coldest) task. There are two important constraints that are satisfied for the recurring thermal profile in the steady state.

- The initial temperature T_{i-1} of a task J_i $(1 \leq i \leq N)$ is the same for all its execution instances in the steady state. This also implies that the final temperature T_i of a task J_i (which is the initial temperature of the next task J_{i+1} in the sequence) is the same for all its execution instances.
- For a single instance of execution of $\mathcal{L}$ in the steady state, the temperature at the beginning of the sequence is identical to the temperature at the end of the sequence, i.e., $T_0 = T_N$.

The **peak temperature of the task sequence** in the steady state is given by $peak(\mathcal{L}) = max(T_1, \ldots, T_N)$. Next we show how to analytically compute $peak(\mathcal{L})$. Following Equation 5 and the constraints on the thermal profile in the steady state, we can express $T_1, \ldots, T_N$ using linear equations.

$$\begin{aligned} T_1 &= (1-m_1)T_{S_1} + m_1 T_N \\ T_2 &= (1-m_2)T_{S_2} + m_2 T_1 \\ &\vdots \\ T_N &= (1-m_N)T_{S_N} + m_N T_{N-1} \end{aligned} \quad (6)$$

This system of N linear equations in N variables $T_1, \ldots, T_N$ can be solved by employing Cramer's rule [20] as

$$T_N = \frac{\left(\begin{array}{l} (1-m_N)T_{S_N} + m_N(1-m_{N-1})T_{S_{N-1}} \\ + m_N m_{N-1}(1-m_{N-2})T_{S_{N-2}} + \cdots \end{array} \right)}{(1 - m_1 m_2 \ldots m_N)}$$

To express T_i for all values of i ($1 \leq i \leq N$), we need to define a new operator $\triangleleft$ that computes the index of the predecessor tasks in the sequence. Note that due to the repeating nature of the task sequence, the predecessor of task J_1 is task J_N. Thus, given a task J_i in the task sequence $\langle J_1, \ldots, J_N \rangle$, $i \triangleleft k$ is defined as the index of the k^{th} predecessor task of J_i. Clearly

$$i \triangleleft k = \begin{cases} i-k & \text{if } k < i \\ N+(i-k) & \text{otherwise} \end{cases}$$

Now the temperature T_i at the end of task J_i can be defined as

$$T_i = \frac{\begin{pmatrix} (1-m_i)T_{S_i} + m_i(1-m_{i\triangleleft 1})T_{S_{i\triangleleft 1}} + \cdots \\ +m_i m_{i\triangleleft 1} \ldots m_{i\triangleleft N-2}(1-m_{i\triangleleft N-1})T_{S_{i\triangleleft N-1}} \end{pmatrix}}{(1-m_1 m_2 \ldots m_N)} \quad (7)$$

The maximum of all the intermediate temperatures is the peak temperature of the sequence, that is,

$$peak(\mathcal{L}) = max(T_1, \ldots, T_i, \ldots, T_N) \quad (8)$$

Now that we have formally defined the peak temperature of a task sequence, we can present the formulation of the task sequencing problem.

B. Problem Formulation

The input to our task sequencing problem is a set of N tasks $J = \{J_1, \ldots, J_N\}$ with execution times $c_1, \ldots, c_N$, average power consumption $P_1, \ldots, P_N$, and the corresponding steady state temperatures $T_{S_1}, \ldots, T_{S_N}$ where $T_{S_i} = P_i \times R + T_{amb}, 1 \leq i \leq N$. Our goal is to construct a sequence of these N tasks that minimizes the peak temperature.

Clearly, given N tasks, there exist $N!$ possible sequences. An optimal solution is the sequence with the minimum peak temperature among the $N!$ possible sequences. An exhaustive search technique can enumerate each of the $N!$ possible sequences, compute the peak temperature for each such sequence (using equations in Section III-A), and then return the sequence with the minimum peak temperature. However, as the number of tasks N increases, the computational complexity of this search technique becomes prohibitive.

Moreover, even a special case of the problem where (a) the task sequence executes only once starting with some initial temperature T_{init}, and (b) the temperature at the end of a task in the sequence depends only on the previous task, finding the optimal sequence that minimizes the peak temperature is still NP-hard. This can be proved by a polynomial reduction from the well known bottleneck traveling salesman problem (Bottleneck TSP), which is NP-hard. Bottleneck TSP problem finds the Hamiltonian cycle in a weighted graph with the minimal weight of the most weighty edge of the cycle. Let us construct a complete weighted graph G with N vertices, where each vertex u maps to a distinct task $task(u)$ and the edge weight between two vertices $u \rightarrow v$ is the temperature at the end of execution of the task sequence $\langle task(u) task(v) \rangle$. Finding an optimal solution to the special case of our problem is equivalent to solving the bottleneck TSP problem on graph G. Thus, even the special case of our problem is NP-hard.

In the next subsection, we present a heuristic to solve the task sequencing problem with the objective of minimizing peak temperature.

C. Task Sequencing Algorithm

Our heuristic for task sequencing is based on the following observation. Equation 7 defines, in the steady state, the temperature after task J_i in the sequence $\langle J_1, \ldots J_i, \ldots, J_N \rangle$. Note that $m_i = e^{-c_i/RC}$.

Algorithm 1 Task Sequencing

Input: Task set $J = \{J_1, \ldots, J_N\}$
1: for $(i = 0, \ldots, N-1)$ $\mathcal{L}_i = J_{i+1}$;
2: **while** $N > 1$ **do**
3: for $(i = 0, \ldots, N-1)$ compute $metric_{\mathcal{L}_i}$;
4: sort $(\mathcal{L}_0 \ldots \mathcal{L}_{N-1})$;
5: for $(i = 0, \ldots, \frac{N}{2} - 1)$ $\mathcal{L}_i = \mathcal{L}_i \bullet \mathcal{L}_{N-(i+1)}$;
6: $N = \frac{N}{2}$;
7: **end while**

Thus $0 < m_i < 1$ and in practice, for all our tasks, m_i varies between 0.2356 and 0.6832 depending on the execution time c_i of the tasks. A closer look at Equation 7 reveals that for task J_i, its execution time c_i (contributing towards m_i) and its steady state temperature T_{S_i} have the maximum influence on the temperature T_i at the end of execution of J_i. This is followed by contribution from its immediate predecessor $J_{i\triangleleft 1}$. The contributions from other predecessors of J_i decrease in the order $J_{i\triangleleft 2}, \ldots, J_{i\triangleleft N-1}$. Based on this observation, what should be the characteristics of a task sequence that minimizes the peak temperature?

First, a task with higher steady state temperature and longer execution time is more likely to produce the peak temperature of a task sequence. We can reduce the temperature at the end of this hot task by choosing a cooler task as its predecessor. Also, a cold task is a better candidate to absorb the temperature impact of execution of a hot task. Therefore, it makes sense to put a cold task as the successor of a hot task. In other words, a good task sequence that minimizes the peak temperature must place tasks with opposite characteristics close to each other to get a balanced thermal profile.

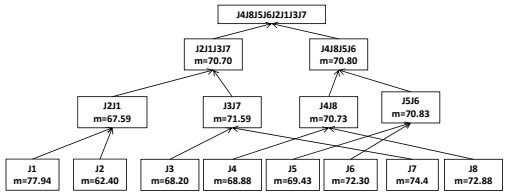

Fig. 3. Task Sequencing Algorithm.

Based on this observation about the characteristics of a good task sequence, we propose a hierarchical algorithm for task sequence construction. Our algorithm proceeds in a bottom-up fashion by pairing up tasks or task subsequences with opposing thermal characteristics till a single task sequence is constructed. Given a set of N tasks, we first pair up tasks with opposite characteristics to create $\frac{N}{2}$ subsequences each containing two tasks. These subsequences are further paired up to create $\frac{N}{4}$ subsequences each containing four tasks. We proceed in this manner till we obtain a single sequence containing all the N tasks. Algorithm 1 details the task sequencing algorithm.

So how do we choose tasks or task subsequences with opposing characteristics? First, we need to define a "metric" that characterizes or summarizes the thermal behavior of a task or a subsequence. Let us first consider individual tasks. Following Equation 5, the temperature at the end of task J_i is defined as $T_i = (1-m_i)T_{S_i} + m_i T_{init}$ where T_{init} is the temperature before execution of task J_i. In a task sequence, however, T_{init} depends on the sequence of tasks executed prior to J_i as shown in Equation 7. As we are constructing the task sequence, T_{init} is unknown. Instead, we approximate the temperature contribution from the other tasks $contrib_{J-\{J_i\}}$ and replace T_{init} with $contrib_{J-\{J_i\}}$ in Equation 5 to get

$$metric_{J_i} = (1-m_i) \times T_{S_i} + m_i \times contrib_{J-\{J_i\}}$$

Here $metric_{J_i}$ summarizes the thermal characteristics of task J_i. We will describe shortly how we approximate $contrib_{J-\{J_i\}}$. But before that let us discuss how we compute *metric* for a a subsequence.

Let $\mathcal{L}$ be a task sequence consisting of a set of tasks denoted by $tasks(\mathcal{L})$. We treat $\mathcal{L}$ as a virtual task with average power consumption and execution time as follows

$$P_{\mathcal{L}} = \frac{\sum_{J_k \in tasks(\mathcal{L})} P_k \times c_k}{\sum_{J_k \in tasks(\mathcal{L})} c_k} \quad \text{and} \quad c_{\mathcal{L}} = \sum_{J_k \in tasks(\mathcal{L})} c_k$$

Thus

$$m_{\mathcal{L}} = e^{-c_{\mathcal{L}}/RC} \quad \text{and} \quad T_{S_{\mathcal{L}}} = P_{\mathcal{L}} \times R + T_{amb}$$

$$metric_{\mathcal{L}} = (1 - m_{\mathcal{L}}) \times T_{S_{\mathcal{L}}} + m_{\mathcal{L}} \times contrib_{J-tasks(\mathcal{L})}$$

Now how do we approximate the thermal contribution of a set of tasks as in $contrib_{J-\{J_i\}}$ or $contrib_{J-tasks(\mathcal{L})}$? We simply set

$$contrib_{J-tasks(\mathcal{L})} = T_{S_{J-tasks(\mathcal{L})}} \quad \text{and} \quad contrib_{J-\{J_i\}} = T_{S_{J-\{J_i\}}}$$

Once we compute this "metric" for individual tasks or subsequences, our pairing strategy is quite straightforward. Let us assume that we have N tasks or subsequences at some level. We sort the tasks or subsequences in decreasing order of *metric* value and pair up the entity in the i^{th} position with the one at $N-(i+1)^{th}$ position, for $i = 0, \ldots, N/2-1$. For ease of exposition, we assume without loss of generality that N is an even number.

When we pair up two tasks or subsequences, the resulting sequence is formed by placing the colder subsequence (lower metric value) before the hotter subsequence (higher metric value). This is based on the observation that the temperature at the end of a task is influenced the most by its predecessor task in the final sequence. An illustration of the working of the algorithm for a task set consisting of eight tasks with sample metric values for the nodes is presented in Figure 3. It can be seen that the algorithm works in a bottom up fashion pairing up tasks or task sequences with opposite characteristics to get the final balanced sequence. The complexity of our task sequencing algorithm is $O(N \times (\lg N)^2)$ where N is the number of tasks.

IV. Sequencing & Voltage Scaling

In this section we explore the possibility of lowering the peak temperature by exploiting voltage scaling or insertion of idle tasks in conjunction with task sequencing. The problem of minimizing peak temperature can be stated formally as follows.

A. Problem Definition

The input to task sequencing and voltage scaling problem are

- A voltage scalable processor having r distinct active states with supply voltages and frequencies $\{(V_1, f_1), \cdots, (V_r, f_r)\}$ where (V_1, f_1) is the highest voltage and frequency level. We also a assume an idle state (*idle*) where no useful work is done and the corresponding power consumption is P_{idle}.
- A set of tasks $\{J_1, \cdots, J_N\}$ with execution times $\{c_1, \cdots, c_N\}$ and power consumptions $\{P_1, \cdots, P_N\}$ where c_i and P_i are the execution time and power consumption of task J_i at the highest frequency f_1 and supply voltage V_1. The power consumption and execution time of task J_i $(1 \le i \le N)$ at active state j $(1 \le j \le r)$ are given by

$$P_{i,j} = \frac{P_i \times V_1^2 \times f_1}{V_j^2 \times f_j} \quad \text{and} \quad c_{i,j} = \frac{c_i \times f_1}{f_j}$$

Algorithm 2 Task Sequencing with Voltage Scaling

Input: Task set $J = \{J_1, \ldots, J_N\}$; deadline D

```
slack = D − Σ_{i=1}^{N} c_i;
for (i = 1,...,N) level(J_i) = 1;
repeat
    L = Task_Sequencing (J);
    Compute peak(L) where J_i is the task with peak(L);
    if c_{i,level(J_i)+1} − c_i ≤ slack then
        slack = slack − (c_{i,level(J_i)+1} − c_i);
        level(J_i) = level(J_i) + 1; c_i = c_{i,level(J_i)}; P_i = P_{i,level(J_i)};
    else
        if min_idle ≤ slack then
            Insert idle task with execution time min_idle into the task set;
            slack = slack − min_idle;
        end if
    end if
until slack > 0
```

- The deadline for one instance of execution of all the tasks where $deadline \ge \sum_{i=1}^{N} c_i$. The slack can be defined as $slack = deadline - \sum_{i=1}^{N} c_i$.

The goal is to produce a task sequence and an assignment of idle times and/or voltages levels to the tasks such that the peak temperature is minimized while satisfying the deadline constraint. We assume that the voltage switching times are negligible in comparison to the execution times of the tasks.

B. Algorithm

Clearly, this problem requires solutions to two mutually dependent sub-problems, namely, voltage assignment and task sequencing. The task sequencing algorithm described in Section III-C takes the power consumption and execution time of the tasks as input, which depend on the active state in which each task executes (voltage assignment). The voltage assignment for peak temperature minimization, in turn, depends on task sequencing as the sequence determines the temperatures reached by the tasks. Therefore, we design an iterative algorithm that repeatedly performs (a) task sequencing to minimize the peak temperature, and (b) voltage assignment to the tasks based on the current sequence so as to lower the peak temperature. The voltage assignment step exploits the "slack" (the difference between the deadline and the total execution time) to lower the voltage/frequency of a hot task or insert idle states to lower the peak temperature. Therefore, the iterative algorithm terminates when $slack = 0$.

Algorithm 2 presents our iterative solution for task sequencing and voltage assignment. Initially, we assume all the tasks are executing at the highest voltage and frequency level (V_1, f_1). We first employ the task sequencing algorithm (Algorithm 1) to return a task sequence $\mathcal{L}$ that minimizes the peak temperature. Given the sequence $\mathcal{L}$, we can determine the peak temperature of the sequence $peak(\mathcal{L})$ in the steady state by solving a system of linear equations as described in Section III-A. Let J_i be the task that produces the peak temperature in the sequence $\mathcal{L}$, that is, the peak temperature is reached at the end of execution of J_i.

Now we proceed to lower $peak(\mathcal{L})$ by exploiting the *slack*. Let $level(J_i)$ be the current voltage and frequency level of task J_i. We first check if we can lower the voltage/frequency level of task J_i by one step to $level(J_i)+1$ and still meet the deadline. If the answer is yes, then the algorithm updates the voltage/frequency level of task J_i, its execution time, power consumption, and the slack. If there is not enough slack to lower the active state of the hottest task J_i, then we introduce an "idle task" with execution time min_{idle} and power consumption P_{idle} to the task set. min_{idle} is the minimum granularity at which we claim the slack time and clearly $min_{idle} \le slack$. We leave the appropriate sequencing of this idle task with respect to the

existing tasks to the next iteration when the task sequencing algorithm is invoked again.

Optimal Voltage Scaling: We compare our approach with [22], which presents a voltage scaling algorithm with a problem formulation closest to ours. However, [22] performs voltage assignment with the objective of minimizing execution time under peak temperature constraint. In contrast, we consider the dual problem of minimizing peak temperature under execution time constraint. We develop an optimal pseudo-polynomial time algorithm (based on dynamic programming) inspired by [22] to solve our voltage assignment problem.

The problem formulation for voltage assignment alone is identical to the formulation discussed in Section IV-A with one major difference. The input is a fixed task sequence $\mathcal{L} = \langle J_1, \cdots, J_N \rangle$ instead of a set of tasks. The goal is to produce an assignment of idle times to the sequence and/or voltage levels to the tasks so as to minimize the peak temperature while satisfying the deadline. Our algorithm is based on the following observation. Given multiple voltage assignments for a sequence of i tasks with the same final temperature and peak temperature, the voltage assignment that results in the smallest total execution time is preferred.

Our dynamic programming algorithm exploits this observation to determine the voltage assignment that minimizes the peak temperature. To incorporate idle times or sleep modes in the formulation, we consider a sequence of $M = 2N+1$ jobs $\langle S_1, \cdots, S_M \rangle$ alternating between original tasks and idle tasks. We develop a recurrence for $E_i(T_{max}, T_f)$ that represents the total execution time corresponding to a voltage assignment and sleep states for the tasks $S_1, \ldots, S_i$, $1 \leq i \leq M$ with maximum observed temperature T_{max} and final temperature T_f. If no such voltage assignment exists, then $E_i(T_{max}, T_f) = \infty$. Note that $E_i(T_{max}, T_f) = \infty$ when $T_{max} < T_f$ as $T_{max} < T_f$ is not feasible. The details of the recurrence equations are not presented here due to space constraints. The algorithm uses the recurrence to compute values of $E_1 \ldots E_M$ for different values of T_{max} and T_f. The optimal solution is the one with the lowest value of T_{max} among all feasible solutions $E_M(T_{max}, T_f)$ where $E_M(T_{max}, T_f) \neq \infty$ and the final temperature at the end of M tasks is less than the initial temperature. The second constraint ensures that the schedule is repeatable.

V. Experimental Evaluation

In this section, we evaluate our thermal management approach. We use the SimpleScalar-3.0 [3] toolset for our experimental evaluation. The power consumptions of the tasks are obtained from Wattch [6], an architectural level power simulator. We use a base supply voltage of 1.2V and a processor frequency of 1.5GHz. For voltage scaling results, we use five different frequency values between 1.5GHz and 800MHz. We model a simple embedded processor architecture resembling ARM Cortex A8 [1]: in-order issue with two integer execution units, a 13 stage pipeline, 32 KB instruction and data caches and a 512 entry branch target buffer.

The temperature values are obtained from HotSpot thermal simulator [19]. The floorplan and silicon area of ARM Cortex A8 processor are provided as input to the thermal simulator. We use the equations and default configuration from HotSpot [13] to compute the thermal resistances and capacitances for our input chip area at each layer. This gives us an RC-network with a thermal resistance of $1.83^oC/Watt$ and capacitance of $112.2 mJoules/^oC$. We use these as the default values for our experiments.

We use a total of 16 benchmarks from MiBench [12] and Media Bench [15] comprising of adpcm, blowfish, crc, dijkstra, djpeg, epic, g721, ghostscript, gsm, lame, mp3, patricia, pegwit, sha, strsearch and susan in this study. The tasks have execution cycles in the range of $4 \times 10^7 - 6 \times 10^8$ cycles and steady state temperatures in the range of $49.85^oC - 88.25^oC$. For our experiments *we create 100 task sets each with eight tasks chosen from the 16 benchmarks*. Some representative task sets are shown in Table I.

Set	Tasks
T1	lame, sha, djpeg, mp3, ghostscript, blowfish, dijkstra, epic
T2	gsm, patricia, adpcm, pegwit, susan, crc, dijkstra, epic
T3	g721, lame, sha, djpeg, pegwit, blowfish, adpcm, lame
T4	gsm, patricia, pegwit, mp3, susan, blowfish, strsearch, epic
T5	lame, g721, ghscript, patricia, blowfish, strsearch, pegwit, sha
T6	gsm, mp3, ghscript, susan, crc, stringsearch, dijkstra, epic
T7	gsm, sha, strsearch, pegwit, mp3, susan, blowfish, patricia
T8	g721, gsm, sha, djpeg, patricia, adpcm, pegwit, strsearch

TABLE I
Representative Task Sets.

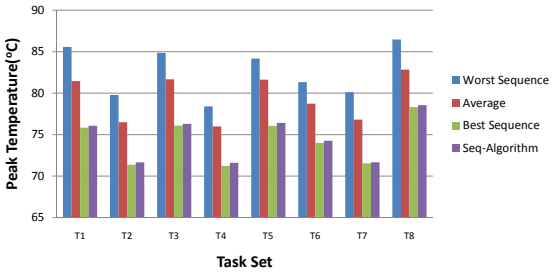

Fig. 4. Accuracy of Task Sequencing Algorithm.

Task Sequencing Algorithm: We evaluate our task sequencing algorithm by comparing the peak temperature of the sequence produced by our algorithm with (a) the peak temperature of the best sequence, (b) the peak temperature of the worst sequence, and (c) the average value of the peak temperature over all possible sequences. For each of the 100 task sets, we generate all 8! (each task set has 8 tasks) possible task sequences and obtain the peak temperature of each sequence through thermal simulation. From these simulation results for each task set, we get the peak temperatures of the best sequence (sequence with lowest peak temperature), the worst sequence (sequence with highest peak temperature) and the expected value of peak temperature (average peak temperature over all possible sequences). Finally, we employ our task sequencing algorithm described in Section III to construct a sequence $\mathcal{L}$ that we expect will minimize the peak temperature. We estimate the peak temperature of the sequence $\mathcal{L}$ returned by our algorithm through thermal simulation.

The results for the eight representative task sets in Table I are shown in Figure 4. Our task sequencing algorithm achieves significantly lower peak temperature compared to the worst sequence and the expected value of peak temperature. More importantly, the peak temperature of the sequence constructed by our algorithm is very close to the peak temperature of the best sequence. The same trends are reflected when we consider all the 100 task sets. When all the 100 task sets are considered, our algorithm has, on an average, a peak temperature 7.47^oC lower than the worst sequence and 4.09^oC lower than the expected value of peak temperature. The peak temperature of the sequence returned by our algorithm is within 0.5^oC of the peak temperature of the best sequence for all the 100 task sets. In the next subsection we examine the impact of voltage scaling techniques on the peak temperature.

Voltage Scaling: We compare the peak temperature returned by our iterative task sequencing and voltage scaling algorithm (Section IV-B) with the optimal voltage scaling algorithm . The result of

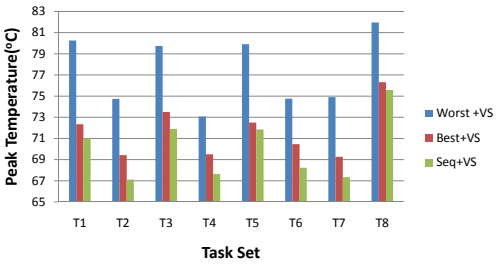

Fig. 5. Advantage of combined sequencing and voltage scaling (Seq+VS) over voltage scaling alone.

the optimal voltage scaling depends on the task sequence provided as input. For each task set, we provide the best and the worst task sequence (at the highest voltage/frequency level) as input to the voltage scaling algorithm. These two, *(Best + VS)* and *(Worst + VS)* represent the best and worst possible scenarios if only voltage scaling is used. The slack is assumed to be 5% of the total execution time of the tasks at highest frequency. The results for the task sets in Table I are presented in Figure 5. Our iterative algorithm that combines voltage scaling and sequencing performs better than even the best possible results with voltage scaling alone. On an average, our algorithm *(Seq + VS)* results in a peak temperature that is 2.1^oC lower than the best scenario *(Best + VS)* and 6.94^oC lower than the worst scenario *(Worst + VS)*. Our algorithm for task sequencing is more efficient than the optimal voltage assignment even though it is iterative in nature. It has a runtime (average 1.45 sec) that is much lower than the runtime of optimal voltage assignment algorithm (average 78.57 sec) while running on 3 GHz Pentium 4 machine with 1 GB memory.

VI. Related Work

Modern low power processors are equipped with on-chip temperature sensors that are monitored continuously. When the temperature exceeds a predefined threshold, different mechanisms are engaged to reduce the temperature. Such mechanisms include voltage and frequency scaling, fetch throttling [5] among others. A comparison of the various hardware based thermal management schemes is presented in [19]. Similarly, software based techniques for thermal management have also been proposed. These include mechanisms such as scheduling priority adjustments [14] and task migrations [10]. These techniques are online or dynamic in nature and are targeted towards general purpose systems.

The well define functionality of embedded systems enables the use of static or design time thermal management techniques. Liu et al. [16] formulate the problem of assigning voltages to tasks on an MPSoC under thermal constraints as a non linear programming problem. They observe that optimizing purely for energy can result in higher peak temperatures. Zhang and Chatha [22] examine the problem of voltage assignments to minimize the total execution time of a set of periodic tasks while operating under thermal constraints. They observe that the voltage mapping problem is NP-hard and develop approximation algorithms. Rao et al. [17] derive the optimal throttling curve to maintain the temperature below a maximum limit. Temperature aware allocation and scheduling in MPSoC for peak temperature reduction have been examined as well. Xie and Hung [21] present an allocation scheme where the maximum temperature is used as one of the factors to drive the allocation of tasks to cores. Chantem et al. [7] formulate the problem of allocation and scheduling on an MPSoC as a mixed integer linear programming problem.

VII. Conclusion

In this paper, we propose task sequencing as a powerful and complimentary mechanism to voltage scaling in thermal management of embedded systems. We propose an efficient algorithm to determine the optimal ordering that minimizes the peak temperature. Our sequencing algorithm achieves a peak temperature that is very close to the peak temperature of the optimal sequence (the difference is less than 0.5^oC). We have also presented an iterative algorithm that combines task sequencing and voltage scaling with the objective of minimizing peak temperature while satisfying the timing constraints. Our combined approach achieves, on an average, $2.1^oC - 6.94^oC$ reduction in peak temperature compared to voltage scaling alone.

VIII. Acknowledgements

This work is partially supported by NUS research project R-252-000-292-112.

References

[1] ARM Cortex A8 Processor. http://www.arm.com/products/CPUs/ARM_Cortex-A8.html.
[2] Failure mechanisms and models for semiconductor devices. *JEDEC publication JEP122C*. http://www.jedec.org.
[3] T. Austin, E. Larson, and D. Ernst. SimpleScalar: An Infrastructure for Computer System Modeling. *IEEE Computer*, 35(2), 2002.
[4] S. Borkar. Design Challenges of Technology Scaling. *IEEE Micro*, 19(4), 1999.
[5] D. Brooks and M. Martonosi. Dynamic Thermal Management for High-Performance Microprocessors. In *HPCA*, 2001.
[6] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A Framework for Architectural-level Power Analysis and Optimizations. In *ISCA*, 2000.
[7] T. Chantem, R. P. Dick, and X. S. Hu. Temperature-aware scheduling and assignment for hard real-time applications on MPSoCs. In *DATE*, 2008.
[8] S. Dropsho et al. Managing static leakage energy in microprocessor functional units. In *MICRO*, 2002.
[9] K. Flautner et al. Drowsy caches: Simple techniques for reducing leakage power. In *ISCA*, 2002.
[10] M. Gomaa, M. D. Powell, and T. N. Vijaykumar. Heat-and-Run: Leveraging SMT and CMP to manage power density through the operating system. In *ASPLOS*, 2004.
[11] S. Gunther et al. Managing the Impact of Increasing Microprocessor Power Consumption. *Intel Technology Journal*, 2001.
[12] M. R. Guthausch et al. Mibench: A free, commercially representative embedded benchmark suite. In *IEEE 4th Annual Workshop on Workload Characterization*, 2001.
[13] W. Huang et al. An improved block-based thermal model in HotSpot 4.0 with granularity considerations. In *Workshop on Duplicating, Deconstructing, and Debunking*, 2007.
[14] A. Kumar et al. HybDTM: A Coordinated Hardware-Software Approach for Dynamic Thermal Management. In *DAC*, 2006.
[15] C. Lee, M. Potkonjak, and W. H. Mangione-Smith. MediaBench: A tool for evaluating and synthesizing multimedia and communicatons systems. In *MICRO*, 1997.
[16] Y. Liu et al. Thermal vs energy optimization for DVFS-enabled processors in embedded systems. In *ISQED*, 2007.
[17] R. Rao et al. An optimal analytical solution for processor speed control with thermal constraints. In *ISLPED*, 2006.
[18] M. T. Schmitz, B. M. Al-Hashimi, and P. Eles. Iterative schedule optimization for voltage scalable distributed embedded systems. *ACM Transactions on Embedded Computing Systems*, 3(1), 2004.
[19] K. Skadron et al. Temperature-aware Microarchitecture: Modeling and Implementation. *ACM TACO*, 1(1), 2004.
[20] G. Strang. *Introduction to Linear Algebra*. Wellesley Cambridge Press, 1993.
[21] Y. Xie and W. L. Hung. Temperature-aware task allocation and scheduling for embedded multiprocessor systems-on-chip (MPSoC) design. *Journal of VLSI Signal Processing Systems*, 45(3), 2006.
[22] S. Zhang and K. S. Chatha. Approximation algorithm for the temperature-aware scheduling problem. In *ICCAD*, 2007.