Abstract: Extracting high performance from multi-core processors requires increased use of dynamic thermal management techniques. This paper presents performance optimal online thermal management techniques for multicore processors. The techniques include dynamic voltage and frequency scaling and task-to-core allocation or task migration. The problem formulation includes accurate power and thermal models, as well as leakage dependence on temperature. The effectiveness of our DVFS and task-to-core allocation techniques are demonstrated by numerical simulations. The proposed task to-core allocation method showed a 20.2% improvement in performance over a power-based thread migration approach.

Keywords: dynamic voltage and frequency scaling, leakage dependence on temperature, makespan minimization, multi-core, performance optimization, task migration.
1. Introduction:
THE TRANSITION to multi-core processors enabled the microelectronics industry to circumvent the power wall— the inability to improve performance of single core processors by simply relying on miniaturization and increasing clock speed, because of the resulting unsustainable growth in power consumption. Designing a package of a future many-core processor to dissipate the maximum possible power consumption will not only be uneconomical but may not be feasible. A more practical solution will be to design the package that is capable of dissipating the average power, and rely on dynamic thermal management (DTM) techniques (dynamic voltage and speed control or DVFS, and task migration) to ensure that the thermal constraints will not be violated when the power dissipation exceeds the capability of the package. Therefore, with manycore processors, DTM techniques are expected to be activated more often than in single core processors.

1.1. Multi Core Processors:
In comparison with single cores, many-core processors will once again exhibit increased power consumption as well as greater temporal and spatial variation in power consumption among the cores. This is due to variations in the number of threads that can be deployed at any time, and the increased intra-core and inter-core process variations due to miniaturization. Designing a package of a future many-core processor to dissipate the maximum possible power consumption will not only be uneconomical but may not be feasible [1], [13].

1.2. Dynamic Thermal Management Techniques:
A more practical solution will be to design the package that is capable of dissipating the average power, and rely on dynamic thermal management (DTM) techniques (dynamic voltage and speed control or DVFS, and task migration) to ensure that the thermal constraints will not be violated when the power dissipation exceeds the capability of the package.
Therefore, with many core processors, DTM techniques are expected to be activated more often (i.e., not just at thermal emergencies) than in single core processors, and consequently, will have a greater negative impact on the performance.

DTM techniques typically control the power and the thermal behavior of a processor by varying three main controls; core speed, core voltage and allocation of tasks to cores. Speed control is the easiest among them and incurs the least performance penalty but offers less power reduction, whereas voltage and task migration offer the greatest power reduction, but suffer from higher performance penalties.

2. Thermal Modeling:

Elevated temperatures impact system reliability, cause difficulties in circuit design and increase the cooling costs significantly [3]. Consequently, accurate thermal modeling has become a requirement. HotSpot is an automated thermal model, which calculates transient temperature response given the physical characteristics and power consumption of units on the die [5]. A fast thermal emulation framework for FPGAs is introduced which reduces the thermal simulation time considerably while maintaining accuracy. As policies based on power metrics are not sufficient to prevent local thermal hot spots, several thermal management approaches have been presented to date. These methods are either dynamic (online) and make decisions while the system is running, or static (offline) where optimizations are performed at design stage. Dynamic thermal management (DTM) controls...
overheating by keeping the temperature below a critical threshold. Computation migration and fetch toggling are examples of DTM techniques [2]. Heat-and-Run performs temperature aware thread assignment and migration for multicore multithreaded systems.

With power density and hence cooling costs rising exponentially, temperature-aware design has become a necessity. Processor packaging is becoming a major expense, and for many chips can no longer be designed for the worst case. Furthermore, simple estimates of power dissipation are not a good proxy for direct measurement or simulation of temperature. There is an urgent need for design techniques to help control or reduce heat dissipation, especially runtime techniques that can regulate operating temperature when the package's capacity is exceeded. Runtime response provides safe cooling and prevents thermal emergencies by changing the processor's behavior rather than relying on costly thermal packaging. Evaluating such techniques, however, requires a thermal model that is practical for architectural studies, especially research or design-space investigations for which no detailed designs are yet available.

### 2.1. Hotspot:

HotSpot is an accurate and fast thermal model suitable for use in architectural studies [5]. It is based on an equivalent circuit of thermal resistances and capacitances that correspond to microarchitecture blocks and essential aspects of the thermal package. The model has been validated using finite element simulation. HotSpot has a simple set of interfaces and hence can be integrated with most power-performance simulators like Wattch. The chief advantage of HotSpot is that it is compatible with the kinds of power/performance models used in the computer-architecture community, requiring no detailed design or synthesis description. HotSpot makes it possible to study thermal evolution over long periods of real, full-length applications.
It is a modeling methodology for developing compact thermal models based on the popular stacked-layer packaging scheme in modern very large-scale integration systems. In addition to modeling silicon and packaging layers, HotSpot includes a high-level on-chip interconnect self-heating power and thermal model such that the thermal impacts on interconnects can also be considered during early design stages. The HotSpot compact thermal modeling approach is especially well suited for preregister transfer level (RTL) and Pre synthesis thermal analysis and is able to provide detailed static and transient temperature information across the die and the package, as it is also computationally efficient.

2.2. Piecewise Linear Approximation:

The piecewise method is similar to the small signal method in that linear network analysis techniques can only be applied if the signal stays within certain bounds. If the signal crosses a discontinuity point then the model is no longer valid for linear analysis purposes. The model does have the advantage over small signal however, in that it is equally applicable to signal and dc bias. In mathematics, a piecewise linear function is a function composed of straight-line sections. It is a piecewise-defined function whose pieces are linear [4]. If the function is continuous, the graph will be a polygonal curve. Important sub-classes of piecewise linear functions include the continuous piecewise linear functions and the convex piecewise linear functions.
Leakage power is given by

\[ P_{l,c,b}(t) = k_1 v_c(t) T_{cb}^2(t) e^{\alpha v_c(t) + \beta} + k_2 e^{\gamma v_c(t) + \delta} \quad (1) \]

\( k_1, k_2, \alpha, \beta, \gamma, \) and \( \delta \) are parameters that depend on circuit topology, size, technology, and design. The non-linear leakage power dependence on temperature and voltage (LDTV), as well as the cyclic dependency between the leakage power and the temperature, complicates the analysis and the solution to various optimization problems.

Without any further simplification, one can only resort to numerical solutions for general non-linear optimization problems. To make any further progress and to develop computationally efficient solutions, this relation needs to be approximated by linear models.

We removed the cyclical dependency between the temperature and the power in equation with the use of PWL.

3. **Dynamic Voltage Frequency Scaling:**

Dynamic Voltage Frequency Scaling (DVFS) is the most commonly used scaling technique. It is very effective to reduce the dynamic power, because the dynamic power is proportional to the voltage squared. The basic goal of DVS is to quickly adjust a microprocessor’s operating voltage to the minimum performance level required by an application. To scale the voltage by a factor of \( \frac{1}{2} \), there would be a 2 savings in power. More aggressively, the voltage supply can even be gated off. In this case, that part of the system will work in a shut-down mode, which can lose all the old states in it. If those states are still needed in future transactions, some recovery logic is needed to restore the component back to the original state. The main drawback of voltage scaling is that the consequent increment in transistor propagation delay, which causes the microprocessor to run at a lower speed. As reported in
previous research, utilizing supply voltages above 3V has little impact on performance. The speed decreases dramatically as $V_{dd}$ approaches the threshold voltage $V_{th}$. Voltage scaling can be used to reduce the static power too.

Because the operating voltage is also related to the operating frequency, the voltage scaling and frequency scaling are tightly coupled. Both techniques can be used both on microprocessors and the memory system. Scheduling support from the operation system is needed to dynamically adjust the operating point according to the application workload. The basic concept behind frequency scaling is that when the clock frequency is lower, there could be less power consumed per cycle. However, the microprocessor has to operate for a longer time to execute the same task.

3.1. Optimal Policy:

The zero-slack policy is just that a policy [10]. It expresses the form of the globally optimal speed function. Here first show that implementation of that policy requires repeatedly solving a convex optimization problem over short intervals. Let $s_{max,c}$ be the speed which sets the temperature of the hottest block of core $c$ at the maximum. Then the optimal speed policy for the core $c$ is given by

$$S^*_c(t) = \begin{cases} 
1, & \text{max}(T_c(t)) < T_{max} \\
0, & \text{max}(T_c(t)) > T_{max}
\end{cases} \quad (2)$$

$$s_{max,c}(t) = \text{max}(T_c(t)) = T_{max}$$

The corresponding voltage is calculated by solving the following equation numerically:

$$S_c(t) = k_v \frac{(v_c(t) - v_{th})^{1.2}}{v_c(t)\text{max}(T_c(t))^{1.15}}, \text{ for all } t, c. \quad (3)$$
The above policy, which we refer to as the zero-slack policy suggests that in order to minimize the overall makespan, either the speed of a core is set to the maximum when the temperatures of all thermal blocks in that core are less than the maximum or the speed should be set such that at least one of the thermal blocks in that core is at the maximum specified temperature.

The above zero-slack policy is just a policy or a guideline to set core speeds and voltages to minimize the makespan. It does not provide a mechanism to implement the policy. In order to find an implementation, we note that the core speeds and voltages determined through the zero-slack policy depend only on the current temperature of the cores and are independent of the core speeds, voltages, and temperatures at any other time instants. Thus, global minimization of makespan can be achieved through local maximization of instantaneous throughput.

3.2. Fast Computational Procedure:

It is a fast computational method, which avoids the complexity of convex optimization [6]. By making use of certain useful properties that matrix $R$ exhibits for scheduling intervals on the order of the die thermal time constant, an efficient computational procedure is developed, whose computational time complexity is linear in the number of cores and logarithmic in the number of discrete states of speeds and voltages. In order to understand the structure of $R$, we need to know the typical multi-core floorplan.

Here we use the floorplan of a dual core Alpha 21264 processor as an example of a typical multi-core floorplan. This floorplan is constructed by replicating the single-core floorplan, such that the processing units of cores are separated by L2 caches. This floorplan is shown to be beneficial as it produces fewer thermal hotspots compared to other floorplans due to
reduced lateral heat flow. The isolation of temperature determination enables the use of binary search technique to determine the speed and the voltage of each core such that they satisfy the optimal policy.

The time complexity of the binary search method is logarithmic in the number of discrete states of speeds and voltages [11].

4. Task Migration:

The basic task allocation problem is to find an allocation of tasks to processors so that all tasks can be scheduled [8]. Depending on the analysis used for testing schedulability, this can also involve discovering suitable task attributes such as priorities. For systems where tasks must communicate with messages, an allocation of message to networks must also be found along with suitable message attributes.

Performing task migration is always a trade-off between response time and reconfiguration overhead. The response time refers to the time between sending the task migration request and actually stopping the execution of a task on the old core while the reconfiguration overhead mainly depends on the amount of data that has to be transferred from the old core to the new core.

4.1. Temperature Aware Task Scheduling:

The doubling of microprocessor performance every 18 months has been the result of two factors: more transistors per chip and superlinear scaling of the processor clock with technology generation [7]. To reduce packaging cost, current processors are usually designed to sustain the thermal requirement of typical workloads and utilize dynamic thermal management (DTM) techniques when temperature exceeds the design-set point. When the operating temperature reaches a predefined threshold, the DTM techniques reduce the processor’s power consumption in order to allow it to cool down. when scheduling a process for execution, the operating system determines on which core the process will run based on the thermal state of each core, that is, its temperature and cooling efficiency. Thermal-aware scheduling is a mechanism that aims to avoid situations such as creation of large hotspots and thermal violations, which may result in performance degradation.

4.2. Linear Assignment Problem:

Thermal-aware scheduling can be implemented purely at the operating system level by adding the proper functionality into the scheduler of the OS kernel. The optimal speeds and voltages of cores for a given allocation can be obtained by setting $T_{\text{die,ij}} = T_{\text{max}}$ and
corresponding \( s_{ij} \leq 1 \). Thus, the computation of speed for a core executing a given task is independent of the speed computations of other cores [11].

Let \( S \) be a matrix, whose elements are \( s_{ij} \), where \( i, j \) refer to the core and the task number, respectively. This matrix is referred to as the speed matrix. Given this speed matrix \( S \), the problem of optimal task-to-core allocation is formulated as given below,

\[
\max_M \sum M'S
\]

(4)

s.t. \( \sum_{i=1}^{n} M(i,j) = 1, j \in \{1, \ldots, q\} \) \hspace{1cm} (5)

\( \sum_{j=1}^{q} M(i,j) = 1, i \in \{1, \ldots, n\} \) \hspace{1cm} (6)

\( M(i,j) \in \{0,1\}, i \in \{1, \ldots, n\}, j \in \{1, \ldots, q\} \) \hspace{1cm} (7)

This is a linear assignment problem and has an efficient polynomial time solution \( O((nq)^3) \) using Munkres algorithm. Once the allocation is determined using this algorithm, the speeds and the voltages within the migration interval are determined for every scheduling interval.

5. Experimental Results:

5.1. Experimental Setup:

A multi-core version of Alpha 21264 processor was used to experimentally validate the proposed techniques. We chose to use the Alpha processor for our experiments as the details of power and thermal models of other processors were not available. The thermal behavior of the processor was modeled using HotSpot-4 thermal-circuit model [9]. The tasks were obtained from MiBench benchmark [12]; their power values were extracted by simulating them using PTScalar [4]. The maximum temperature was limited to 110 °C and the maximum frequency of operation was set at 2 GHz. The supply voltages of cores were allowed to vary from 0.3 V to 1.2 V. The total dynamic power of the processor was restricted to 230 W, while the leakage power contribution was limited to 60 W. The convectional thermal resistance in the HotSpot thermal model set at 0.35 °C/W. The scheduling interval for the voltage–speed scaling method was fixed at 10 ms. The migration interval was fixed at 100 ms to minimize the computation overhead.
5.2. Optimal Makespan Minimization Policy:

Fig. 6. shows the results of execution of the optimal makespan policy (Section III) for four tasks executing on four cores. The temperature plots are for the hottest block in each core. Hence the speeds are set to the maximum. As the temperatures increase, the speeds of the cores are throttled exponentially to maintain the core temperatures at the maximum. We also see that once a task is completed, it enables other tasks to execute at faster pace as there is less power consumption.

5.3. Effect of Cores and Tasks on the Performance of Task-to-Core Allocation:

Fig. 7 shows the performance of the proposed optimal task allocation for various combinations of number of tasks and cores against the P.TM scheme for similar configuration. The plot shows the percentage improvement in throughput. Higher task-to-core ratio provides more flexibility for the proposed algorithm to extract the maximum throughput in comparison with the DVFS technique.
5.4. Performance Improvement through Task Migration:

![Figure 8: Throughput without Migration](image)

This section demonstrates the advantages of task migration in improving the overall performance by comparing the throughput improvement by executing a set of tasks with and without task migration. Fig. 8 and Fig. 9 show the plot of throughput of a four-core processor executing four tasks chosen out of a set of eight tasks for a duration of 200 s with the initial package temperature set at 35°C. Throughput is measured as the sum of core speeds weighted by their respective IPCs.

![Figure 9: Throughput with Migration](image)

First, we execute tasks according to the optimal voltage–speed scaling alone for a fixed allocation of tasks to cores. Next, we execute the same set of tasks using both the task migration and the optimal voltage–speed scaling. We see there is an improvement in throughput with the task migration, hence demonstrating the need for task migration to improve performance.
6. Conclusion
Developing efficient online techniques for multi-core processors is necessary for maximum processor utilization under changing workload conditions. In this paper, the optimal DVFS scheduling problem was addressed as separate problems of task-to-core allocation over migration intervals and voltage– speed scaling within migration intervals. Practically implementable techniques for both the problems were developed and shown to be optimal. Simulations demonstrated that our techniques achieved significant performance improvements over the existing techniques.
References:


