Hybrid Dynamic Energy and Thermal Management in Heterogeneous Embedded Multiprocessor SoCs

Shervin Sharifi
CSE Department,
University of California, San Diego
shervin@ucsd.edu

Ayse Kivilcim Coskun
ECE Department
Boston University
acoskun@bu.edu

Tajana Simunic Rosing
CSE Department,
University of California, San Diego
tajana@ucsd.edu

ABSTRACT- Heterogeneous multiprocessor system-on-chips (MPSoCs) which consist of cores with various power and performance characteristics can customize their configuration to achieve higher performance per Watt. On the other hand, inherent imbalance in power densities across MPSoCs leads to non-uniform temperature distributions, which affect performance and reliability adversely. In addition, managing temperature might result in conflicting decisions with achieving higher energy efficiency. In this work, we propose a joint thermal and energy management technique specifically designed for heterogeneous MPSoCs. Our technique identifies the performance demands of the current workload. By utilizing job scheduling and voltage/frequency scaling dynamically, we meet the desired performance while minimizing the energy consumption and the thermal imbalance. In comparison to performance-aware policies such as load balancing, our technique simultaneously reduces the thermal hot spots, temperature gradients, and energy consumption significantly.

1. INTRODUCTION

Heterogeneous multicore architectures can have significant power and performance advantages in comparison to homogeneous systems [10]. A heterogeneous MPSoC consists of cores with various power and performance characteristics integrated on the same die. Assigning applications to these cores, such that runtime resource requirements of the applications fit the core characteristics, leads to higher overall computational efficiency and lower power consumption than homogeneous MPSoCs [24]. In this paper, we propose a novel job scheduling method integrated with energy management to reduce the thermal challenges while saving energy and meeting the performance demand.

High temperatures cause a number of challenges. In addition to increasing the cooling costs, thermal hot spots accelerate failure mechanisms such as electromigration, stress migration, and dielectric breakdown, which cause permanent device failures [9]. In fact, more than 50% of all integrated circuit failures are related to thermal issues [16]. High temperatures degrade performance as well, as the effective operating speed of devices decreases with increasing temperature. Also, leakage power, which constitutes a considerable portion of the total power consumption in today's chips, is exponentially related to temperature. Similar to thermal hot spots, large temperature variations on the die affect reliability and performance. Thus, addressing thermal hot spots alone is not enough to achieve high reliability, and at moderate temperatures temperature gradients determine device reliability [11]. Temperature variations across the chip can also lead to performance degradation and timing problems by influencing gate and wire delay characteristics. Therefore, thermal hot spots and gradients should be handled jointly to mitigate the problems related to reliability, performance, leakage, and cooling cost.

Well-known techniques for temperature management include clock-gating, dynamic voltage and frequency scaling (DVFS), and task migration, which come with a performance cost[16]. To reduce temperature-induced problems in homogeneous MPSoCs with a low impact on performance, previous work has introduced temperature-aware workload scheduling and efficient task migration techniques ([5], [13]). To the best of our knowledge, the impact of dynamic thermal and power management techniques in heterogeneous architectures has not been addressed before. The capability of heterogeneous MPSoCs to off-load workload from more complex hotter cores to simpler low-power cores create new opportunities for thermal management. Moreover, the workloads in embedded systems are likely to be heterogeneous, i.e., with various performance requirements and power consumption values. Therefore, for such workloads, heterogeneous MPSoCs propose a better fit by allowing customizing the power and performance characteristics of the system for the current application. However, performing thermally-aware and performance efficient mapping of the workload on the MPSoC is a non-trivial problem. In addition, temperature variations as high as 50ºC across the die are reported in modern microprocessors [16],[34]. Spatial thermal variations are typically larger in heterogeneous MPSoCs due to the intrinsic disparity in power densities across the chip. As traditional power and thermal management techniques do not necessarily address thermal balancing, management techniques for heterogeneous systems should specifically take the thermal gradients into consideration. Also, embedded systems usually use cheaper cooling infrastructures, and they rely more on natural convection for removing the heat due to tighter cost and area constraints. This makes high temperatures and large temperature variations even more important challenges for such systems.

In this work, we propose a joint temperature and energy management solution for heterogeneous embedded MPSoCs. When the system is under-utilized, our technique focuses on maximizing the energy savings, while maintaining the temperature at safe levels. For workloads with a higher performance demand, we apply a thermal balancing policy specifically designed for heterogeneous MPSoCs. Our technique includes an offline workload profiling phase, where we characterize the possible incoming tasks. Then, for various levels of performance demands, we compute the voltage/frequency settings that balance the temperature across the chip. At runtime, we integrate DVFS (which utilizes the offline analysis) with a temperature- and performance-aware task assignment strategy. When the required performance can be provided by only some of the cores in the MPSoC, our technique selects which cores to turn off to increase the energy savings. By considering the workload requirements and the power consumption characteristics of the system simultaneously, we achieve low and balanced thermal profiles. Our experimental results on a heterogeneous MPSoC designed based on commercial cores show that our technique achieves significant reduction in hot spots and spatial gradients, and reduces energy consumption by 12% in comparison to performing load balancing and clock-gating, while improving the performance by over 20%.

The rest of this paper starts with a discussion of the related work in Section 2, and we provide the experimental methodology in Section 3. Section 4 explains the details of our novel thermal management technique. We provide the experimental results in Section 5, and conclude the paper in Section 6.

2. RELATED WORK

Dynamic thermal management methods either control temperature by reducing the power consumption, or by re-distributing the workload on the available units on the chip. Temperature-aware clock-gating [26] and DVFS [14] are two hardware mechanisms used in reducing temperature at the cost of performance loss. Temperature-aware scheduling is a method for managing temperature with reduced performance cost. A temperature-aware scheduling method for single-core processors is proposed in [19]. When a thermal emergency occurs, the jobs which cause high temperatures are identified and penalized by limiting the time slice assigned by the scheduler. In [20], authors propose selective scheduling of available integer and floating point threads in a simultaneously multi-threaded processor to prevent hot spots in the register files.

When there are redundant units available on the system, ping-ponging activities among these identical units improves the thermal profile significantly [21]. The technique proposed in [22] takes advantage of already existing redundancy in the pipeline resources in superscalar processors, and balances the power density by controlling the utilization of issue queues, register files, and ALUs. Heat-and-Run performs temperature-aware thread migration for multicore multi-threaded systems [7]. In [4], the trade-offs among various hot spot mitigation schemes, thermal time constants, and workloads are investigated on a POWER5.

So far, activity migration has been mostly used among identical resources, and there is limited prior work on activity migration among heterogeneous resources. At the microarchitecture level, [17] proposes a thermally-aware superscalar microprocessor with a secondary simpler lower power pipeline. When temperature exceeds a given threshold, the main superscalar pipeline is clock-gated while the secondary in-order pipeline is activated. Asymmetric dual core design [18] includes an extra low-power core, and offloads workload to this core to reduce the occurrence of thermal emergencies at a low performance impact.

Utilizing a combination of software and hardware approaches provides the opportunity to adjust the thermal management response according to the severity of thermal emergency. For thermal management of homogeneous MPSoCs, the technique in [5] uses thread migration along with per-core DVFS to mitigate the hotspots and reduce temporal and spatial variations. In [6], the authors evaluate various combinations of thread migration, DVFS, and clock gating in a homogeneous MPSoC, and they use control theoretic techniques for closed loop control. In [10] and [24], the authors have studied performance and power advantages of heterogeneous MPSoCs over their homogeneous counterparts.

To the best of our knowledge, thermal behavior of heterogeneous systems has not been studied before. In [25], the authors argue that the benefits of heterogeneous MPSoCs can be increased by the use of dynamic scheduling policies. They experiment their dynamic thread assignment policies on a combination of Alpha EV5 and Alpha EV6 processors running SPEC2000 benchmark suite. However, their work has focused only on the performance and has not studied any power or temperature effects. The statistical techniques proposed in [28] optimize task scheduling in heterogeneous MPSoCs for makespan, and they achieve significant improvement over approaches that consider worst-case characteristics. However, these approaches do not consider energy or temperature.

In this work, we propose an energy and temperature management technique for heterogeneous MPSoCs. Our technique utilizes task scheduling and voltage/frequency scaling for reducing hot spots, minimizing temperature gradients, and saving energy while meeting the performance requirements.

3. METHODOLOGY

The embedded systems workloads usually cover a wide spectrum of power and performance requirements. Therefore, for such systems, heterogeneous multicore systems offer a good match for providing the desired performance at reduced power consumption. The cores we use in our experiments are:

1. A low-power in-order architecture similar to the SPARC cores in UltraSPARC T1 [12];
2. A very low power core designed for embedded systems, similar to Intel's XScale [33].

Similar to most of the embedded processors, these cores are in-order processors. Power, performance, and area characteristics of these cores are shown in Table 1. We assume that the MPSoC is implemented in 65 nm technology. The areas of the cores are derived from published photos of the dies after subtracting the area occupied by I/O pads, interconnection wires, interface units, L2 cache, and control logic as in [24], and scaled to 65nm. We also include on-chip L2 caches in our simulations. Each L2 cache has 1MB size, 2 banks, 64-byte lines, and is 4-way associative. Using CACTI [21], the area and power consumption of the caches at 65nm are estimated as 14mm^2 and 1.7W, respectively. The cache power consumption value includes leakage.

A. Power and Performance Modeling of Cores

Our methodology decouples performance and power simulation of each core type from the overall system simulation. We accomplish this by collecting the performance and power data for all the benchmarks for the different types of cores using the M5 Simulator [1]. This modular framework allows for an easy extension of the set of cores simulated in the heterogeneous MPSoC, and it is capable of integrating a variety of simulators or real-life experiments if needed. The system architecture can be assumed as either single-ISA or multiple-ISA in this set-up. In this work, we assume a multiple-ISA architecture, where pre-compiled binaries are available for each core type. The only limitation in the multiple-ISA architecture is regarding runtime job migrations. As the execution time of each application is fairly short (see Section 3.B.), we do not stall the execution and migrate, but instead alter the job allocation policy.

Table 1. Characteristics of the cores

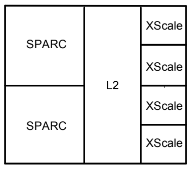
Processor		XScale	SPARC
Issue Width		1	2
I-Cache		32KB	32KB
D-Cache		32KB	32KB
Area (mm^2)		1.4	5
Frequency Settings	1.2V	624MHz	1.2GHz
	1.18V	416MHz	1.14GHz
	1.06V	208MHz	1.0 GHz
Average Dynamic Power (W)	1.2V	0.34	1.9
	1.18V	0.32	1.8
	1.06V	0.28	1.6

The first core type we use is similar to a single-threaded version of SPARC. This core has a local simple branch predictor, issue width of 2 instructions, and 32KB instruction and data cache sizes [12]. The second core is based on the architecture and power characteristics of very low power embedded cores, such as XScale. We assume the same three voltage settings for the XScale and SPARC cores. For XScale, we use the existing available frequency levels (as reported in [33]), and for SPARC we set the default frequency to 1.2GHz (as reported in [12]), and scale frequency using the 95% and 85% settings as in [8]. To model the in-order pipeline of SPARC and XScale, we modify M5's execution engine and forced the instructions to operate in-order. We utilize Wattch [3] for the power modeling of the cores and update the model parameters for 65nm. We directly integrate Wattch with M5 to provide dynamic power measurements for each application. We observe that for each core, the dynamic power values have little variation for our benchmark set. Thus, we compute the average power consumption values at each voltage/frequency setting for each of the cores. These power values are then utilized in the temperature simulations. We compute the leakage power of CPU cores based on structure areas and temperature. For the 65nm process technology, we use a base leakage power density of 0.5W/mm^2 at 383K [2]. For transitions during frequency scaling, we use the power of the higher power state, and we assume idle power during clock gating. We set idle power to 1/3 of average dynamic power. We compute temperature dependence using the model introduced in [21] with the same constants mentioned in the paper for 65nm. The overhead of switching to a new frequency is $500\mu s$ [30], [31]. The overhead for clock gating is 36000 CPU cycles of Xscale [32].

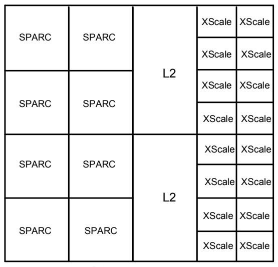
B. Workload

The workloads in this paper consist of a set of heterogeneous tasks arriving at different rates to the system. To create a set of workloads representing a broad range of applications, we use various mixes of integer benchmarks provided in MiBench [23]. MiBench suite has benchmarks from automotive/industrial, network and telecommunications applications.

We use two heterogeneous MPSoCs in our experiments which are shown in Figure 1. MPSoC1 consists of 2 SPARC-like cores and 4 XScale-like cores, and is more suitable for cellular applications. MPSoC2 has 8 SPARC-like cores and 16 XScale-like cores, and fits better to applications such as wireless base stations. We run workloads that combine several MiBench benchmarks for 100 seconds on each MPSoC. For MPSoC1, we combine network and automotive benchmarks (bitcount, basicmath, qsort, dijkstra and patricia), while for MPSoC2, we run telecommunication and network benchmarks (qsort, dijkstra, patricia, adpcm and fft). We vary the system utilization for each MPSoC between medium to high for each workload. We change the arrival rates of the tasks during the execution to test the system with various performance demands; i.e., utilizing 45% to 95% of the processing capability of the MPSoC. This way, the workloads for both MPSoCs provide a wide range of runtime scenarios.



(a) MPSoC1



(b) MPSoC2

Figure 1 Heterogeneous MPSoC used in our experiments

C. Thermal Model

We use HotSpot Version 4.2 [14] as the thermal modeling tool. We run the thermal simulations with a sampling interval of 10 ms, which provides a good precision. The operating system scheduler ticks in modern OSes (e.g., Linux) occur every 10ms as well. We initialize HotSpot with steady state temperature values. The parameters used in HotSpot are summarized in Table 2.

Table 2. HotSpot Parameters

Die Thickness	0.15mm
Convection Capacitance	140J/K
Convection Resistance	1.5 K/W – 4 K/W
Heat Sink Thickness	1 mm
Spreader Thickness	0.1 mm

We calculate the die and package characteristics based on the trends reported for 65nm process technology for an embedded system package. In embedded systems such as cell phones there is no heat sink or spreader. To simulate this, we set the heat sink/spreader thickness to 1mm, and 0.1mm in HotSpot respectively [27]. For MPSoC1, due to lack of a good heat sink and cooling, we set convection resistance to 4K/W. For MPSoC2, we use a convection resistance of 1.5 which represents better packaging and cooling. Thermal convection resistances up to 20K/W have been reported in [29].

4. ENERGY AND THERMAL MANAGEMENT FOR HETEROGENEOUS MPSOCS

In this section we present our joint temperature and energy management technique for heterogeneous MPSoCs. At runtime, our technique has two different modes. At low and moderate utilization levels, the system works in its *normal mode,* where the goal is to maximize the energy savings while meeting the performance demands and thermal constraints. At *higher utilization mode*, where high temperatures and temperature variations are more likely to happen, the system switches to thermal balancing, which prevents local concentration of high power densities. Although saving energy is still important in this mode, since high temperatures and temperature variations are likely to cause performance and reliability degradation, thermal management has a higher priority.

The two decisions to make in either mode are: (1) where to allocate an incoming task, and (2) which voltage/frequency (VF) setting each core should be running at. In both *normal* and *high utilization* modes, task assignment is performed based on task characteristics, which are collected by an offline task characterization step. In normal mode, since thermal emergencies are less likely to happen, the cores control their own VF setting to meet the performance demand of the current workload. In the high utilization mode, since the autonomous control of frequency settings without a global knowledge of the temperature profile can be thermally unsafe, the core VF settings are decided at the global level. This global frequency assignment uses a table of pre-computed VF settings. In the table, each entry has a thermally safe VF configuration (consisting of a VF setting for each of the cores) for a given performance demand. At runtime, based on the workload, the most appropriate VF configuration to meet the current processing demand is selected. As noted above, offline phase of our method consists of two parts: (1) task characterization, and (2) thermally safe frequency setting calculations. The results of these steps are used at runtime for directing task assignment, and also for the frequency assignment in high utilization mode. Next, we provide the concepts that will be used in explaining the details of our technique, and discuss the offline and online phases in detail.

Overview:

To ensure that the scheduling and DVFS maintains the desired performance level, we measure "processing demand" at runtime for the current workload, and adjust the "processing capability" of the MPSoC in an energy- and temperature-aware way to meet this demand. *Processing demand* is measured by the number of instructions arriving per second to the MPSoC. *Processing capability* is the number of instructions the MPSoC executes in a second. This metric allows combining the processing capability of various cores into a single measure which can be compared against the workload demand. *VF setting* represents the voltage/frequency setting of a core. *VF configuration* of the MPSoC is the set of all VF settings applied to the cores of MPSoC. The arriving tasks are accumulated in queues, and they wait for getting assigned to cores for execution.

Length of a queue refers to the number of instructions in the queue to be executed. The length of queue Q_j is calculated as:

$$length(Q_j) = \sum_{y_i \in Q} m_i \qquad (1)$$

where $y_i \in Q$ refers to the tasks y_i that are in the queue Q, and m_i is the number of instructions in task y_i.

Phase 1: Offline Phase

The offline phase consists of task characterization and the computation of VF configurations used in the *high utilization mode*. In this subsection, we provide the details of each step.

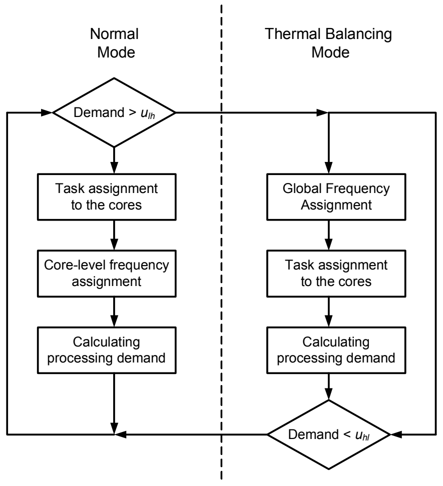

Normal Mode
Thermal Balancing Mode
Demand > u_{lh}
Task assignment to the cores
Core-level frequency assignment
Calculating processing demand
Global Frequency Assignment
Task assignment to the cores
Calculating processing demand
Demand < u_{hl}

Figure 2. Runtime operation of the technique

A. Task Characterization

The power and performance of an application vary depending on what type of core the application is allocated on, and also depending on the application's characteristics. For example, when a memory intensive task is allocated on a multiple-issue core running at higher frequency, the performance gain might not be noticeable in comparison so running the same task on a single-issue core. On the other hand, allocating a CPU intensive task in a higher performance core can result in significant performance benefits. Therefore, assigning tasks to the right cores is crucial to maximize performance benefits. During offline task characterization, we investigate the tasks which are expected to run on the system to quantify their execution characteristics on the various core types available on the heterogeneous MPSoC. In this step, each target task is run alone on each core type at every available VF setting. Performance and power characteristics of the benchmarks such as runtime, IPC, power consumption, and number of instructions are recorded in a table. This information is used for identifying which core type is the most suitable for running each task. In our work, since we are interested in optimizing energy consumption, we choose energy consumption as the parameter to decide on the suitability of a task for a core. So for each task, we order the cores based on the energy consumption while running the task. This information is used at runtime task assignment to direct the tasks to their most suitable (i.e., lowest energy) cores when possible.

B. VF Configuration Calculation for Thermal Balancing

The goal of this step is to determine the VF configurations that balance the temperature across the chip. In other words, this step selects the VF configurations resulting in thermal profiles with the least magnitude of spatial variations. We use HotSpot [14] thermal model which uses a thermal RC network to simulate the thermal behavior at the architectural level. Using this model, at the steady state, we have:

$$T = RP \tag{2}$$

where T and P are the vectors of temperature values and power consumptions on the cores, and R is a matrix calculated based on the thermal resistances in the network. R is extracted based on the floorplan and package [14]. If the number of cores on MPSoC is n, T and P are vectors of length n, and R is an $n \times n$ matrix.

The goal of our thermal balancing algorithm is achieving similar temperatures across the chip. Targeting a constant temperature value, T_{all}, for all of the cores, we rewrite Eqn. (2) as:

$$P = R^{-1} \times \begin{bmatrix} T_{all} \\ \vdots \\ T_{all} \end{bmatrix} = R^{-1} \times T_{all} \times \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} = R^{-1} \times T_{all} \times 1_n \tag{3}$$

where 1_n is a vector of length n whose elements are all 1. The vector P contains the target power value for all cores to balance the temperature across the chip, and it is computed as:

$$P_i = T_{all} \sum_{j=1}^{n} R^{-1}_{i,j} \tag{4}$$

The target VF setting for each core i is the setting with the closest power consumption value less than or equal to P_i. Setting each core i to frequency $f_i(T_{all})$ for target temperature T_{all}, the "processing capability" of the MPSoC at temperature T_{all} (denoted with $C(T_{all})$) is estimated by the following equation:

$$C(T_{all}) = \sum_{i=1}^{n} \left(IPC_i^{f_i(T_{all})} \times f_i(T_{all}) \right) \tag{5}$$

where $IPC_i^{fi(Tall)}$ is the average IPC of core i at frequency $f_i(T_{all})$ over the characterized tasks. Recall that IPC is calculated during the offline task characterization step. $C(T_{all})$ is the average number of instructions the MPSoC is able to execute per second, and this is the metric we utilize for processing capability. Thus, Eqn. (5) provides a mapping of the processing capability of the MPSoC to the temperature which MPSoC operates at, and then to the corresponding VF settings to achieve that balanced temperature. All of these values, i.e., corresponding processing capability, temperature, and VF settings of different cores for that temperature, are stored in a lookup table indexed by the processing capability. At runtime, based on the estimated processing demand of the incoming workloads, the cores are set to the corresponding VF settings based on the table. The size of the table generated is the number of VF configurations times the number of cores. Runtime of the Matlab implementation for VF configuration calculation is in the order of seconds in our experiments.

Phase 2: Runtime Phase

We need to make decisions to dynamically adjust the system configuration to meet the performance demand when necessary. At runtime, our scheduler assigns jobs to the cores and sets the core frequencies at regular time intervals. So, one of the two loops in Figure 2 is iterated at each scheduler tick. To decide between normal mode and high utilization modes, we define *processing demand thresholds u_{lh} (low to high)* and u_{hl} (high to low) as shown in Figure 2. The two thresholds for switching between these modes prevent oscillation between these two modes. Next we describe the online phase in detail.

A. Task Assignment

We assume a hierarchical queue distribution shown in Figure 3. This hierarchy allows modular control over the system by decoupling the decisions made at various levels. It also provides scalability of this approach to many-core MPSoCs. The levels of hierarchy in our system include MPSoC level, core-instance level and core-type level. At MPSoC level, the incoming tasks are distributed among different core types based on their suitability (i.e., energy efficiency, as discussed in Section III.B) for different core types. Defining other suitability criteria is possible as well.

If the initial distribution of tasks among different types of cores cannot be met due to restrictions such as power, temperature or performance, the tasks originally assigned to a core type would be redirected to the alternative core type.

At the next level (i.e., core-type level), the tasks are distributed among different instances of the same core type based on individual settings of the cores and also on the current workload. When distributing the tasks at the core-type level, if instances of a core type cannot provide enough processing capability for the demand of the incoming workload, tasks originally sent to a core type will be redirected to the other core types.

As shown in Figure 2, in normal mode, first the task assignment is performed, and then each core matches its frequency to its individual workload demand (as will be explained later). While in high utilization mode, first the global VF configuration of MPSoC is set based on the processing demand of the workload, and then the tasks are assigned to the cores such that the queue length is maintained close to l_{target}. l_{target} is set to the product of maximum processing capability of the core and a constant γ (in our experiment γ=3) divided by scheduling interval.

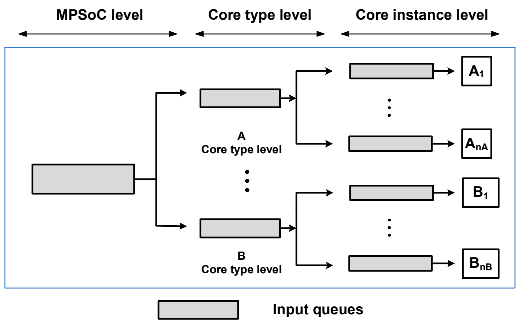


Figure 3 Task distribution hierarchy

B. Frequency assignment

The multi-level queue structure shown in Figure 3 provides useful information about the processing capability and processing demands. Changes in the length of a queue indicate how well the processing capability at that level is matched to the demand of the incoming workload. A shrinking queue length shows over-provisioned processing capability for the current demand. Likewise, a growing queue length indicates under-provisioned processing capability for the current demand. A constant queue length means that the provided processing capability matches the processing demand.

To adjust the frequency of a core to the processing demand in normal mode, we use the information in the core's queue. If we assume constant arrival rates between times τ and τ', Eqn. (6) represents the dynamics of the queue.

$$l(\tau') - l(\tau) = (\lambda(\tau) - \mu(\tau)) \times (\tau' - \tau) \qquad (6)$$

where $\lambda(\tau)$ and $\mu(\tau)$ represent the arrival rate of the tasks and service rate of the queue. The length of the queue is known at times τ and τ'. Moreover, if τ and τ' are two consecutive scheduling ticks, we know that the frequency setting of the core doesn't change in between. Thus, we estimate the processing rate of the core at frequency f_i to be $\mu_{fi}(\tau)=IPC^{s}_{fi}\times f_i$ where IPC^{s}_{fi1} is the average IPC of suitable tasks for this core when running on it at frequency f_i. We calculate the arrival rate of the tasks for this core as:

$$\lambda(\tau) = \mu_{f_i}(\tau) + \frac{l_{f_i}(\tau') - l_{f_i}(\tau)}{(\tau' - \tau)} \qquad (7)$$

We estimate the change in the length of the queue if the core was running at a different frequency f_j using Eqn. (8):

$$l_{f_j}(\tau') - l_{f_j}(\tau) = (\lambda(\tau) - \mu_{f_j}(\tau)) \times (\tau' - \tau) \qquad (8)$$

Therefore, we estimate the changes in the length of the queue for other frequency settings with a very low computation overhead. The frequency resulting in the closest queue length to the l_{target} in N recent intervals would be chosen as the proper frequency setting for the core. In other words, at time τ, we look at the values of l_{f_j} during previous N intervals. For different frequencies, the absolute error between l_{target} and potential queue lengths at other frequencies is calculated as follows:

$$e_{f_i}(\tau) = \sum_{k=0}^{N} \left| l_{target} - l_{f_i}(\tau - k) \right| \qquad (9)$$

f_i producing the minimum e_{fi} will be chosen as the current frequency setting. If the queue length is still decreasing even for the lowest available frequency, the core will be turned off. The total processing demand of the workload is computed as:

$$d(\tau) = \sum_{k=0}^{n} \lambda_k(\tau) \qquad (10)$$

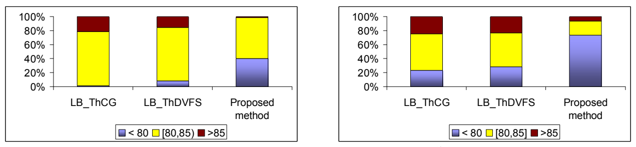
where $\lambda_k(\tau)$ is the processing demand of core k at time interval τ. The total demand is used for switching between two modes. The calculated processing demand is also used in high utilization mode to access the lookup table for VF configurations.

5. EXPERIMENTAL RESULTS

In this section, we evaluate our technique in terms of temperature, energy, and performance. In our experiments, we utilize the MPSoCs and workloads defined in Section 3. We compare our proposed method with the following well-known techniques: (1) load balancing combined with clock-gating (*LB_ThCG*), (2) load balancing combined with temperature-triggered DVFS (*LB_ThDVFS*). For task assignment, both policies run load balancing, which balances the number waiting tasks among the available cores. In *LB_ThCG*, when the temperature threshold is exceeded on a core, clock is gated to reduce the power consumption and temperature. *LB_ThDVFS* reduces the frequency and voltage of a core when it reaches a critical threshold. In both policies, a core with a thermal emergency is switched back to operating at the default (highest frequency) state when the temperature is sufficiently low.

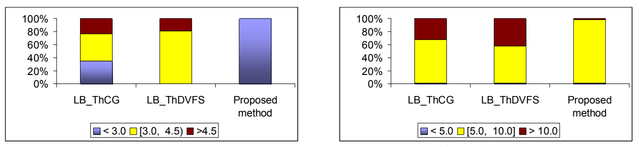
In our experiments, the critical thermal threshold is set at 88°C. We assume the maximum safe temperature is 90°C, and we set the activation threshold for the dynamic management techniques slightly below 90°C to prevent hot spots. To decide that a core is cool enough to switch back to the default operation mode, we use another lower threshold value of 85°C. We utilize this second threshold to prevent oscillatory behavior when a core is operating close to 88°C. Figure 4 shows the distribution of hot spots across the chip, averaged over all the cores. We observe that our technique completely eliminates the hot spots larger than 90°C, and reduces the percentage of time spent above 85°C more than four times for MPSoC1.

Figure 5 shows the distribution of spatial temperature variations. As shown in this figure, our technique lowers the frequency of high temperature variations, and achieves a more even thermal distribution across the die. We normalized the energy and performance results with respect to *LB_ThCG*. Figure 6 compares energy consumption of all three techniques. Our technique improves the energy consumption by 8% and 12%, respectively, in comparison to *LB_ThCG* and *LB_ThDVFS*. The lower energy consumption is achieved mainly due to our energy-aware task assignment scheme in normal mode. The energy consumption is further reduced by even distribution of heat across the chip in high utilization mode, which reduces the hot spots and leakage power. Figure 7 provides a comparison of performance, which is measured by the average finishing time of the tasks. Our technique has up to 10% and 20% better performance than *LB_ThDVFS* and *LB_ThCG* respectively.



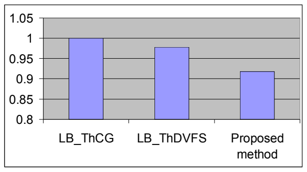
(a) MPSoC1 (b) MPSoC2

Figure 4 Distribution of hotspots HotSpots (°C)

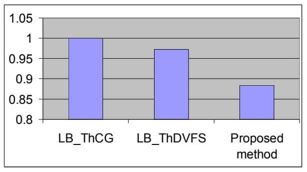


(a) MPSoC1 (b) MPSoC2

Figure 5 Spatial temperature variations (°C)

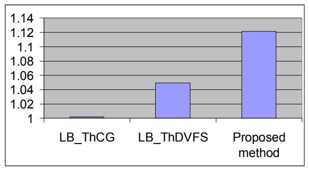


(a) MPSoC1

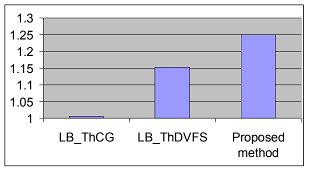


(b) MPSoC2

Figure 6 Normalized energy consumption



(a) MPSoC1



(b) MPSoC2

Figure 7 Normalized performance

The improved performance is a result of the heterogeneity-aware task assignment to different core types. Also, as our technique achieves lower thermal profiles without stalling and slowing down the cores during thermal emergencies, we avoid most of the performance penalty in the other techniques.

6. CONCLUSION

Heterogeneous MPSoCs offer better performance-energy trade-offs in comparison to homogeneous MPSoCs, especially for embedded systems that run highly heterogeneous workloads. However, as such heterogeneous systems integrate various cores with different power characteristics; they have the potential to create large spatial thermal gradients. In addition, workload scheduling to achieve the desired performance while saving energy is a significant challenge on such systems. In this paper, we have proposed a novel technique to perform efficient runtime management of heterogeneous MPSoCs. Our technique maintains the temperature at safe levels, reduces the spatial thermal gradients, and improves energy savings while still meeting the desired performance. We distinguish between low and high system utilization levels, and combine voltage/frequency scaling and job scheduling to address the current performance demand. We also utilize offline analysis of tasks running on the MPSoC, and computation of thermally-safe voltage/frequency settings to improve the efficiency of the runtime management. Our technique improves the energy savings by 12%, and reduces thermal hot spots and spatial gradients by orders of magnitude in comparison to load balancing while maintaining the target performance.

Acknowledgements

This work has been funded by NSF Project GreenLight grant 0821155, NSF SHF grant 0916127, GSRC grant 2003-DT-660, UC Micro grant 08-039, Texas Instruments, Sun Microsystems, and Cisco.

REFERENCES

[1] N. L. Binkert, E. G. Hallnor, and S. K. Reinhardt. "Network-oriented full-system simulation using M5." CAECW, 2003.
[2] P. Bose. "Power-efficient microarchitectural choices at the early design stage." Keynote Address, PACS, 2003.
[3] D. Brooks, V. Tiwari, and M. Martonosi. "Wattch: a framework for architectural-level power analysis and optimizations." ISCA, 2000.
[4] J. Choi, et. al. "Thermal-aware task scheduling at the system software level." ISLPED, 2007.
[5] A. K. Coskun, T. Rosing, and K. Whisnant. "Temperature aware task scheduling in MPSoCs." DATE, 2007.
[6] J. Donald and M. Martonosi. "Techniques for multicore thermal management: Classification and new exploration." ISCA, 2006.
[7] M. Gomaa, M. D. Powell, and T. N. Vijaykumar. "Heat-and-Run: leveraging SMT and CMP to manage power density through the operating system." ASPLOS, 2004.
[8] C. Isci, G. Contreras, and M. Martonosi. "Live, runtime phase monitoring and prediction on real systems with application to dynamic power management." MICRO 39, 2006.
[9] Failure mechanisms and models for semiconductor devices, JEDEC publication JEP122C. http://www.jedec.org.
[10] R. Kumar, D. M. Tullsen, P. Ranganathan, N. P. Jouppi, and K. I. Farkas. "Single-isa heterogeneous multi-core architectures for multi-threaded workload performance." ISCA, 2004.
[11] C. J. Lasance. "Thermally driven reliability issues in microelectronic systems: status-quo and challenges." Microelectronics Reliability, 43:1969– 1974, 2003.
[12] A. S. Leon et al., "A Power-Efficient High-Throughput 32-Thread SPARC Processor," *IEEE JSSCC,* vol.42, no.1, Jan. 2007
[13] S. Murali, A. Mutapcic, D. Atienza, R. Gupta, S. Boyd, L. Benini, and G. D. Micheli. "Temperature control of high-performance multi-core platforms using convex optimization." DATE, 2008.
[14] K. Skadron, M. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan. "Temperature-aware microarchitecture." ISCA, 2003.
[15] D. Tarjan, S. Thoziyoor, and N. P. Jouppi. CACTI 4.0. Technical Report HPL-2006-86, HP Laboratories Palo Alto, 2006.
[16] M. Pedram and S. Nazarian, "Thermal modeling, analysis, and management in VLSI circuits: Principles and Methods," Proc. of IEEE, Special Issue on Thermal Analysis of ULSI, Vol. 94, No. 8, pp. 1487-1501, 2006.
[17] C.-H., Lim, W., Daasch, AND G. Cai, "A thermal-aware superscalar microprocessor." ISQED, 2002.
[18] S. Ghiasi and D. Grunwald, "Design Choices for Thermal Control in Dual-Core Processors," Workshop on Complexity-Effective Design, 2004.
[19] E. Rohou and M. Smith, "Dynamically managing processor temperature and power," Workshop on Feedback directed optimization, November 1999.
[20] J. Donald and M. Martonosi. "Leveraging Simultaneous Multithreading for Adaptive Thermal Control." Workshop on Temperature-Aware Computer Systems, 2005.
[21] S. Heo, K. Barr, and K. Asanovic. "Reducing power density through activity migration," ISLPED, pp. 217–222, 2003.
[22] M.D. Powell, E. Schuchman, and T.N. Vijaykumar, "Balancing resource utilization to mitigate power density in processor pipelines," MICRO'38, 2005.
[23] M. R. Guthaus, et. al., "MiBench: A free, commercially representative embedded benchmark suite." IEEE Annual Workshop on Workload Characterization, 2001.
[24] Rakesh Kumar, et. al., "Processor Power Reduction Via Single-ISA Heterogeneous Multi-Core Architectures," Computer Architecture Letters, Volume 2, Apr. 2003
[25] Michela Becchi and Patrick Crowley. "Dynamic thread assignment on heterogeneous multiprocessor architectures." Journal of Instruction-Level Parallelism, Vol. 10, pp. 1-26, June 2008.
[26] S. Gunther, et. al., "Managing the impact of increasing microprocessor power consumption." Intel Technology Journal. 2001.
[27] M. Datta, T. Osaka, J. W. Schultze, Microelectronics Packaging, CRC Press. 2005.
[28] N.R. Satish, K. Ravindran, and K. Keutzer. "Scheduling task dependence graphs with variable task execution times onto heterogeneous multiprocessors." International Conference on Embedded Software, 2008.
[29] B. Vandevelde, et. al., "Characterization of the polymer stud grid array (PSGA), A lead free CSP for high performance and high reliable packaging", SMTA, 2001.
[30] P. D'Alberto, M. Puschel and F. Franchetti, "Performance/energy optimization of DSP transforms on the XScale processor." HIPEAC, 2007.
[31] K. Choi, R. Soma, and M. Pedram. "Dynamic voltage and frequency scaling based on workload decomposition." ISLPED 2004.
[32] A. Shrivastava, E. Earlie , N. Dutt , Alex Nicolau, "Aggregating processor free time for energy reduction," CODES-ISSS, 2005.
[33] Intel PXA270 processor, electrical, mechanical and thermal specification data sheet. http://www.intel.com
[34] T. Sato, et. al., "On-chip thermal gradient analysis and temperature flattening for SoC design", ASP-DAC, 2005