# Temperature-Aware Runtime Power Management for Chip-Multiprocessors with 3-D Stacked Cache

Kyungsu Kang[1] and Giovanni De Micheli[2]
[1] CAE Team, Samsung Electronics, Hwasung, Korea
[2]LSI, EPFL, Lausanne, Switzerland

Seunghan Lee and Chong-Min Kyung
EE, KAIST, Daejeon, Korea

***Abstract*—The advent of 3-D fabrication technology makes it possible to stack a large amount of last-level cache memory onto a multi-core die to reduce off-chip memory accesses and, thus, increases system performance. However, the higher power density (i.e., power dissipation per unit volume) of 3-D integrated circuits (ICs) might incur temperature-related problems in reliability, leakage power, system performance, and cooling cost. In this paper, we propose a runtime solution to maximize the performance (i.e., instruction throughput) of chip-multiprocessors with 3-D stacked last-level cache memory, without thermal-constraint violation. The proposed method combines runtime cache tuning (e.g., cache-way partitioning, cache-way power-gating, cache data placement) with per-core dynamic voltage/frequency scaling (DVFS) in a temperature-aware manner. Experimental results show that the integrated method offers 23% performance improvement on average in terms of instructions per second (IPS) compared with temperature-aware runtime cache tuning only.**

## I. Introduction

3-D integration is an emerging technology that can stack multiple active silicon layers on top of each other, and connect them through vertical interconnects [e.g., through silicon vias (TSVs)]. For instance, a large amount of last-level cache can be stacked onto a multi-core die to reduce the latency and power consumption resulting from off-chip memory accesses [1][2]. As more cache memory tiers are stacked, the increased cache capacity is expected to improve the system performance owing to the reduced number of off-chip memory accesses. However, multiple (memory) die stacking may cause a drastic increase in power density which causes temperature-related problems in reliability (e.g., negative bias temperature instability, time-dependent dielectric breakdown), power consumption (i.e., temperature-induced leakage power), performance (i.e., increased circuit delay as temperature increases), and system cost (e.g., cooling and packaging cost).

For power and thermal management techniques in 3-D chip multiprocessors (CMPs), dynamic voltage frequency scaling (DVFS) is a well-known technique to control power consumption of processing cores and, thus, the whole system temperature [3][4][5]. DVFS takes an advantage of the fact that linear reduction in the supply voltage can quadratically reduce the power consumption, while linearly slows down the clock frequency. When operating temperature approaches the thermal limit, reducing core's voltage/frequency effectively reduces the operating temperature, but inevitably leads to system performance degradation. In conjunction with core's thermal management, facilitating thermal management techniques for stacked memory also needs to be considered because temperature of the stacked memory (and the whole system temperature) substantially increases owing to the heat transferred from the processing cores as well as the heat generated by the stacked memory itself. Turning off a subset of cache memory (e.g., cache ways) can suppress the cache leakage power, thus help reduce the whole system temperature. However, too aggressive power-gating of cache resources may incur performance degradation due to the increase in cache misses. Thus, the amount of cache capacity should be determined at runtime according to the memory access demand of applications running on 3-D CMPs [6].

In this paper, we propose a runtime solution that integrates both DVFS and runtime cache tuning in a temperature-aware manner to maximize performance of 3-D CMPs in terms of instructions per second (IPS), while keeping the operating temperature from the maximum temperature limit. Although temperature-aware DVFS and cache tuning are self-sufficient techniques, applying them together help fully exploit the potential of both techniques. For example, under a given temperature (and/or power) limit, one can reduce core's supply voltage to activate more cache resources (i.e., to allocate more cache capacity to the processing cores) to reduce off-chip memory accesses. On the other hand, one can turn off more cache resources and increase core's supply voltage and clock speed. For the cache tuning techniques, way-based cache partitioning [7] and power-gating [8] are used. The contributions of this paper are as follows.

1) We simultaneously considered way-based cache partitioning, power-gating, and per-core DVFS to maximize performance of 3-D CMPs in a temperature-aware manner. We developed analytical models that help determine the positions of cache ways to be turned off, the allocation of cache ways to each core, and the supply voltage of each core in order to maximize the instruction throughput without violating the temperature limit. The analytical models exploit the inherent thermal characteristics of 3-D ICs, i.e., heterogeneous cooling efficiency (e.g., silicon layers closer to the heat sink have higher cooling efficiency) and heterogeneous thermal coupling (e.g., blocks in different dies but in the same horizontal position have a strong thermal correlation among them).

2) Based on runtime workload monitoring units, we applied the proposed analytical models as a runtime solution with considerations of memory access behavior of applications (i.e., CPU stall time induced by off-chip memory accesses). For fast computation, some numerical algorithms and a look-up table (LUT) are used. Experimental results show that the integrated method (i.e., the proposed one) gives a significant performance improvement than applying either technique without the other.

A solution that simultaneously manages the power consumption of cores and cache memory was first suggested in [25], where a runtime greedy algorithm is proposed to find an energy-minimal cache and voltage/frequency configuration among some pre-defined configurations (i.e., combinations of available core's voltage/frequency level, cache line size, and cache set-associativity). In [26], a runtime solution for single core is proposed that dynamically adopts the core's voltage/frequency and the cache configuration parameter (e.g., the number of cache ways) according to the workload requirements (e.g., number of execution cycles, number of L2 cache accesses/misses, etc.). J. Zhao et al. [27] applied voltage control to both processor cores and MRAM-based L2 cache hierarchy where a fixed cache capacity is assigned to cores, which is likely to lead to suboptimal result. However, [25], [26], and [27] have considered neither peak temperature nor temperature-induced leakage energy, which is not applicable to 3-D CMPs. Kang et al. [28] first proposed a thermal management technique that simultaneously manages the power consumption of cores and stacked cache memory in 3-D CMPs as a design-time solution, not applicable at runtime.

The rest of this paper is organized as follows. Section II explains preliminaries. Section III gives a motivational example of our work. Section IV presents the problem definition. Section V presents analytical formulations to solve the defined problem. Section VI explains how to apply the analytical formulations at runtime. Section VII presents the experimental setup and results followed by conclusion in Section VIII.

## II. Preliminaries

### A. Heterogeneous Thermal Characteristics in 3-D CMPs

Fig. 1 illustrates a coarse-grained thermal model of a 3-D CMP where each block (e.g., core and cache) is represented by a set of thermal model element (i.e., thermal resistance, heat capacitance, and current source) [4][5]. In Fig. 1, the heat sink is located at the bottom of the die stack. The power consumption of a block influences the temperature of other blocks as well as its own temperature. However, vertically adjacent blocks have much larger thermal influence on each other than horizontally adjacent blocks, which is called *heterogeneous thermal coupling*. It is because heat dissipation occurs mostly in the vertical direction due to much larger thermal resistance between horizontally adjacent blocks (i.e., $R_{intra}$) than that between vertically adjacent blocks (i.e., $R_{inter}$). According to the experimental data in [4], $R_{intra}$ is approximately 2.44K/W, while $R_{inter}$ approximately 0.15K/W, i.e., $R_{intra} \sim 16 \cdot R_{inter}$, which means that most heat propagates vertically.

Ignoring the heat flow in horizontal direction, the steady-state temperature of each block is calculated as follows.

$$T_3 = P_3 \cdot R_{inter} + T_2 \tag{1}$$

$$T_2 = (P_2 + P_3) \cdot R_{hs} + T_{amb} \tag{2}$$

where $T_2$ and $T_3$ are the temperatures of Block 2 and Block 3, $P_2$ and $P_3$ are the power consumed by Block 2 and Block 3, respectively. $T_{amb}$ is the ambient temperature and $R_{hs}$ is the thermal resistance from Block 2 to the ambient through the heat sink. As shown in Eqns. (1) and (2), blocks closer to the heat sink have higher cooling efficiency than those located farther from the heat sink because the heat generated from the farther blocks is transferred to the heat sink through more thermal resistors than the closer blocks, which we call *heterogeneous cooling efficiency*. Details about these heterogeneous thermal characteristics with experimental data are also shown in [4][5][9].

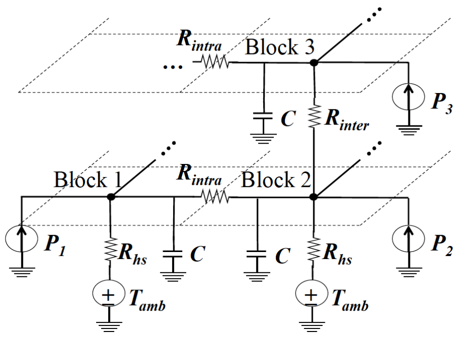

Fig. 1. Coarse-grained thermal model of a general 3-D CMP where each block has uniform temperature distribution.

### B. Application's Execution Time

A task can be defined as a sequence of instructions to be executed. Thus, execution time of a task is sum of latencies of all instructions in the task, which can be divided into two parts; on-chip and off-chip latencies. The on-chip latency is caused by events that occur inside the core such as data dependency, cache hit, branch prediction, etc. These events can be synchronized to the internal clock and the resultant delay can be reduced by increasing core's clock frequency. The off-chip latency, on the other hand, is not affected by changing the core's clock frequency since accesses to off-chip memory (resulting from last-level cache misses) are synchronized to the memory bus clock. Thus, execution time of a task, $t^{ex}$ can be presented as follows.

$$t^{ex} = \frac{w}{f} + t^{stall} \tag{3}$$

where $w$ is the number of core clock cycles when there is no core stall for off-chip memory accesses, $f$ core clock frequency, and $t^{stall}$ stall time spent by a core for the completion of the off-chip memory accesses. Note that the amount of change in the execution time owing to the clock frequency change is limited to the first part of the right-hand side in Eqn. (3). Fig. 2 illustrates execution time of three programs, gzip [10], equake [10], and free_cfd [11]. *gzip* shows the most significant performance improvement as the core clock frequency increases, while *equake* shows little performance improvement. It is because *equake* is memory bound that there are many off-chip memory accesses owing to the last-level cache misses. The execution time of such a *memory-bound application* is relatively independent of the core clock frequency.

## III. Motivation

In this section, we explain the motivation of our work with an example of 3-D CMPs. Fig. 3 shows a 3-D CMP consisting of two cores with eight tiers of stacked L3 (last-level) cache memory. Each cache tier consists of eight cache ways each

TABLE I. RESULTS OF THE MOTIVATIONAL EXAMPLE

| Scheme | # of cache ways assigned to Core 1 | # of cache ways assigned to Core 2 | # of cache ways stacked on Core 1 | # of cache ways stacked on Core 2 | Clock frequency (GHz) of Core 1 | Clock frequency (GHz) of Core 2 | Normalized IPS |
|---|---|---|---|---|---|---|---|
| DCA only | 8 | 5 | 7 | 6 | 3.00 | 3.00 | 1.00 |
| DCA with DVFS | 25 | 13 | 18 | 20 | 2.35 | 1.68 | 1.24 |
| DCAP with DVFS | 22 | 14 | 24 | 12 | 1.59 | 2.23 | 1.30 |

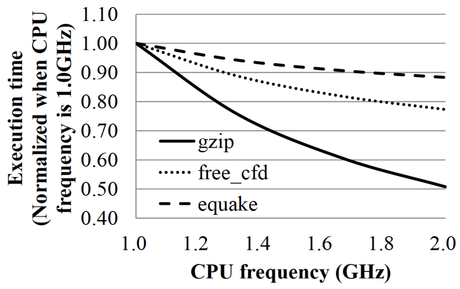

Fig. 2. Execution time of three applications (i.e., gzip, equake, and free_cfd) with respect to core clock frequency.

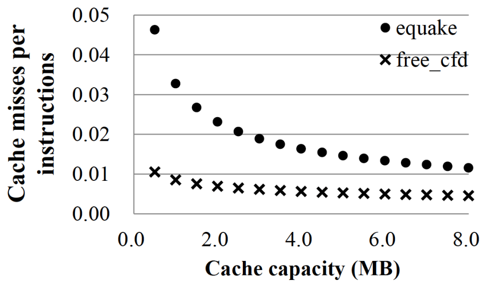

Fig. 4. Cache misses per instruction for *equake* and *free_cfd* with varying L3 cache size from 256KB to 8MB while the core clock frequency is fixed.

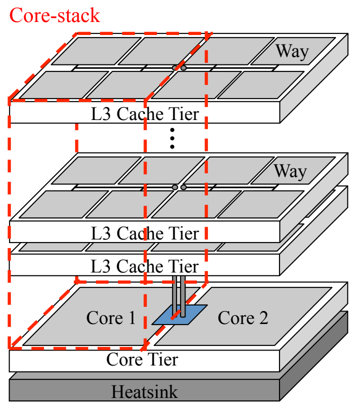

Fig. 3. A motivation example of 3-D CMPs consisting of two cores with eight tiers of L3 cache.

of which has 256KB capacity. The core tier is located next to the heat sink since cores are major heat sources. Let us assume that *equake* and *free_cfd* are mapped to Core 1 and Core 2, respectively. The average power consumed by cores running *equake* and *free_cfd* are, respectively, 30.65W and 61.87W at 2GHz clock frequency. Fig 4 shows cache misses per instruction for *equake* and *free_cfd* as L3 cache capacity varies from 256KB to 8MB when two programs are executed separately. The target 3-D CMP, which is constrained by maximum temperature, $T_{max}$=80$^oC$, dynamically performs per-core voltage/frequency scaling and per-cache way power gating during runtime. The clock frequency (and the corresponding supply voltage) varies from 1GHz (1.097V) to 3GHz (1.372V). As explained in Section II, a core and cache ways directly stacked on the core have strong thermal correlation. Thus, we call the core and the cache ways in the same stack a *core-stack* as shown in Fig. 3.

Table I shows performance results of three different thermal management schemes [i.e., dynamic cache-way assignment (DCA) only, DCA with DVFS, and dynamic cache-way assignment and placement (DCAP) with DVFS] in terms of instructions per second (IPS). In Table I, the second column shows the average numbers of cache ways *logically* assigned to Core 1 and Core 2, respectively. The third column shows the average number of activated (i.e., turned on) cache ways stacked on Core 1 and Core 2, respectively. The fourth column shows the average clock frequency of Core 1 and Core 2, respectively. The instructions per second (IPS) results are normalized with respect to that of *DCA only*.

In case of *DCA only*, cache ways are assigned to Core 1 and Core 2 without per-core voltage/frequency scaling such that the sum of IPS is maximized while the temperature of each block (i.e., core and cache way) does not exceed $T_{max}$. As shown in Table I, more cache ways are assigned to Core 1 than Core 2, because *equake* (mapped onto Core 1) achieves larger performance improvement than *free_cfd* (mapped onto Core 2) as the assigned cache capacity increases, as shown in Fig. 4. In *DCA only* scheme, we assume that as many physical cache ways are turned on as required to be assigned to each core, starting from the heat sink side because of the heterogeneous cooling efficiency (e.g., silicon blocks farther from the heat sink have higher temperatures.). As the result, the physical distribution of activated cache ways are more even across Core-stack 1 and Core-stack 2 than their logical assignment as shown in Table I. However, the performance of *DCA only* is limited by the relatively small number of assigned cache ways because, without voltage/frequency scaling, the number of activated cache ways needs to be lowered to reduce the whole power consumption thereby meeting the temperature constraint.

In our first method, i.e., *DCA with DVFS*, if the marginal performance gain due to the cache capacity allocation is larger than that due to core's frequency assignment while the power consumption due to either method remains equal, then more cache ways are activated and assigned to the cores while lowering the supply voltage and clock frequency of cores in order to satisfy the maximum temperature limit, as shown in Table I. Compared with *DCA only*, *DCA with DVFS* allows 19% performance improvement. Note that we still assume that the cache ways, which are logically assigned to each core, are turned on starting from the tier closest to the heat sink.

While, *DCA with DVFS* focuses only on the effectiveness of cache capacity assignment with voltage/frequency scaling of cores, *DCAP with DVFS* further enhances the performance by considering both application's memory boundedness and heterogeneous thermal coupling of 3-D CMPs together. In *DCAP with DVFS*, more cache ways are activated in the core-stack running application with higher memory boundedness because the execution time of the core (running the application with higher memory boundedness) is less sensitive to the change of core clock frequency. Thus, as shown in Table I, *DCAP with DVFS* activates more cache ways in Core-stack 1 than Core-stack 2, although the number of cache ways logically assigned each core is similar to that of *DCA with DVFS*. Since more cache ways are activated in Core-stack 1, more power consumed by cache ways is incurred there. Thus, in order to meet the temperature constraint in Core-stack 1, Core 1's clock frequency is set to a lower level, on the other hand, Core 2's clock frequency is set to a higher level without violating the temperature limit because the number of turned on cache ways in Core-stack 2 is reduced. This frequency setting in *DCAP with DVFS* gives additional performance improvement because memory-bound application (e.g., *equake*) is less sensitive to frequency change than CPU-bound application (e.g., *free_cfd*), as shown in Table I.

## IV. Problem Definition

This paper focuses on a multi-core system where 3-D L3 cache is stacked as shown in Fig. 5. The tier closest to the heat sink consists of multiple cores with its own private L1 and L2 cache. Multiple tiers of L3 cache each of which consists of multiple cache ways are stacked on the multi-core tier. Each core dynamically changes its clock frequency and voltage level at a discrete time interval. Each cache way can be dynamically turned on and off when it is necessary. The discrete time intervals for both scaling of voltage/frequency and turning on/off of cache ways are determined by the scheduler in the operating system (OS). System temperature and performance profiling, e.g., instructions per cycle (IPC), are obtained from temperature sensors and hardware performance counters at runtime as they are provided in most modern processors. The delay overhead of accessing temperature sensors and hardware performance counters is assumed to be negligible.

The problem is to find 1) frequency/voltage of each core and 2) cache way placement (i.e., assigning cache ways to each core and turning off unassigned cache ways) at every discrete time interval, e.g., 200ms, such that the instruction throughput of the target 3-D CMP is maximized while the temperature constraint is met. The problem can be defined as follows. Given the number of cores, $M$, an allocated thread set, the number of stacked cache tiers, $L$, and the total number of cache ways, $B$, the problem is to find the clock frequencies (and the corresponding voltages) of each core, $f_{set} = f_1, f_2, ..., f_M$, the number of cache ways assigned to each core, $b_{set} = b_1, b_2, ..., b_M$, and the number of power-on cache ways that are directly stacked on each core, $l_{set} = l_1, l_2, ..., l_M$ such that the total sum of IPS (instructions per second), $IPS_{sum}$ is maximized while keeping the temperature of each core and cache way within the given temperature limit, $T_{max}$. The problem is represented as follows.

$$\textbf{Find} \quad f_{set},\ l_{set},\ b_{set} \tag{4}$$

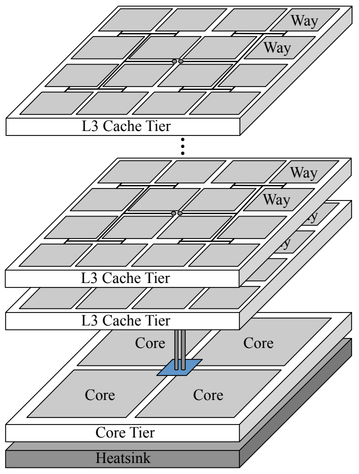

Fig. 5. An example of target 3-D CMPs consisting of four cores with 3-D stacked L3 cache. Each core has its own private L1 and L2 cache that are not depicted in this figure.

$$\textbf{such that} \quad IPS_{sum} = \sum_{i=1}^{M} IPS_i \quad \textbf{is maximized} \tag{5}$$

$$\textbf{subject to} \quad T_i^{core} \leq T_{max} \quad ; \forall i = 1, 2, ..., M \tag{6}$$

$$T_j^{way} \leq T_{max} \quad ; \forall j = 1, 2, ..., B \tag{7}$$

$$\sum_{i=1}^{M} b_i = \sum_{i=1}^{M} l_i \leq B \tag{8}$$

$$l_i \leq \frac{W \cdot L}{M} \quad ; \forall i = 1, 2, ..., M \tag{9}$$

where $IPS_i$ is IPS of core $i$, and $W$ is the number of cache ways per tier.

## V. Analytic Formulations for Cache Tuning with Voltage/Frequency Scaling

In Section V-A, we explain how to determine the performance-maximal number of cache ways assigned to each core, i.e., $b_{set}$ when the total number of activated (i.e., turned on) cache ways, $B_{active}$ is given. In Section V-B, we explain how to determine the positions of cache ways to be turned on as well as the clock frequency of each core such that the performance degradation owing to the decrease in clock frequency is minimized. In Section V-C, we explain how to determine the total number of activated cache ways, $B_{active}$ that maximizes the system performance.

### *A. Cache Way Assignment to Cores*

The number of cache misses ($N_i^{miss}$) decreases as a power law of the amount of cache capacity as follows [13].

$$N_i^{miss}(b_i) = \gamma_i^{ref} \cdot (C_{way} \cdot b_i)^{-\mu_i} \tag{10}$$

where $C_{way}$ is the cache capacity per way and $\gamma_i^{ref}$ is the reference cache miss ratio which depends on the application running on core $i$. The exponent, $\mu_i$ also depends on core $i$ and typically lies between 0.3 and 0.7. To estimate $\gamma_i^{ref}$ and $\mu_i$, the cache monitoring circuit proposed in [7] is adopted.

In Eqn. (3), $t^{stall}$ can be modeled by the number of last-level cache misses as follows [14].

$$t_i^{stall} = \alpha \cdot N_i^{miss} + \beta \tag{11}$$

where $\alpha$ and $\beta$ are empirical constants determined by system memory architecture (e.g., bus/memory frequency and width). The rationale of modeling $t^{stall}$ only with the number of last-level cache misses is two-fold. First, the effect of last-level cache miss dominates the others (e.g., TLB miss, interrupts, etc.). Second, the number of hardware counters simultaneously monitored in a processor is usually limited.

Based on Eqns. (10) and (11), the stall time of core $i$, $t_i^{stall}$ is represented as a function of cache capacity assigned to core $i$ (i.e., $b_i$), therefore $IPS_{sum}$ in Eqn. (5) can be rewritten as a function of $b_i$, as follows.

$$IPS_{sum}(b_i) = \sum_{i=1}^{M} \frac{N_i}{t_i^{ex}(f_i, b_i)} = \sum_{i=1}^{M} \frac{N_i}{\frac{w_i}{f_i} + t_i^{stall}(b_i)} \tag{12}$$

where $N_i$ is the total number of instructions executed in core $i$. When the clock frequency of each core (i.e., $f_i$) is fixed, $IPS_{sum}$ in Eqn. (12) is a non-decreasing concave function with respect to the number of assigned cache way, $b_i$. Thus, we introduce the concept of performance improvement (PI) of each core with respect to the number of assigned cache ways as follows.

$$PI_i = \frac{\partial IPS_i}{\partial b_i} \tag{13}$$

where $PI_i$ indicates the IPS improvement of core $i$ due to the increase in cache capacity by one additional cache way. $IPS_{sum}$ can be maximized when the total number of activated cache ways, $B_{active}$ is distributed to all the cores such that each core achieves the same PI as follows.

$$PI_1 = PI_2 = ... = PI_M \tag{14}$$

A proof of Eqn. (14) can be given by using *Lagrange function* [15], which is omitted due to the page limit.

## B. Cache Way Activation with Voltage/Frequency Scaling

As described in Section II-A, a core and cache ways within the same core-stack have strong thermal correlations with each other. That is, the temperature of a core-stack depends mainly on its own power consumption (consumed by the core and the cache ways in the same core-stack). We refer to cache ways which are stacked on core $i$ and positioned on the $j^{th}$ cache tier as *way-set* ($i$, $j$). Based on Eqns. (1) and (2), the temperatures of core $i$ and way-set ($i$,$j$) can be presented as follows.

$$T_i^{core} = R_{hs} \cdot (P_i^{core} + \sum_{j=1}^{L} P_{i,j}^{way-set}) + T_{amb} \tag{15}$$

$$T_{i,j}^{way-set} = R_j \cdot \sum_{k=j}^{L} P_{i,k}^{way-set} + T_{i,j-1}^{way-set} \tag{16}$$

where $P_i^{core}$ and $P_{i,k}^{way-set}$ are, respectively, the power consumptions of core $i$ and way-set ($i$, $k$) and $R_j$ is the thermal resistance between the way-set on tier $j-1$ and the way-set on tier $j$. [Tier 0 indicates the core tier in Eqn. (16)].

According to Eqns. (15) and (16), the temperature of top tier, $T_{i,L}^{way-set}$ becomes the highest temperature of core-stack $i$. As the number of activated (i.e., turned on) cache ways in core-stack $i$ increases, $T_{i,L}^{way-set}$ [shown in Eqn. (16)] increases more than linearly with respect to the number of activated cache ways in core-stack $i$ (i.e., $l_i$) and frequency of core $i$ (i.e., $f_i$) needs to be reduced when the maximum temperature of core-stack $i$ exceeds the temperature limit, $T_{max}$. Thus, $f_i$ can be represented as a function of $l_i$. If the number of cache ways assigned to each core, $b_i$ is fixed, Eqn. (12) can be rewritten as follows.

$$IPS_{sum}(l_i) = \sum_{i=1}^{M} \frac{N_i}{\frac{w_i}{f_i(l_i)} + t_i^{stall}} \tag{17}$$

$IPS_{sum}$ in Eqn. (17) is a non-increasing concave function with respect to $l_i$ because the maximally allowable clock frequency of core $i$, which is obtained when $T_{i,L}^{way}$ becomes $T_{max}$, decreases more than linearly with respect to $l_i$. Thus, we introduce the concept of performance loss of each core, $PL_i$ with respect to $l_i$ as follows.

$$PL_i = \frac{\partial IPS_i(l_i)}{\partial l_i} \tag{18}$$

where $PL_i$ indicates the IPS loss of core $i$ owing to the decrease of clock frequency resulting from one additional activated cache ways in core-stack $i$. Thus, in order to maximize $IPS_{sum}$ in Eqn. (17), the number of activated cache ways in each core-stack needs to be determined such that each core has the same performance loss per cache way activated as follows.

$$PL_1 = PL_2 = ... = PL_M \tag{19}$$

We also omit the proof of Eqn. (19) because the proof is basically the same as Eqn. (14) using *Lagrange function*.

## C. Total Number of Activated Cache Ways Decision

Fig. 6 shows the result of the sum of IPS of each core with respect to the total number of activated cache ways, $B_{active}$, when a set of benchmark applications are mapped on cores and cache way assignment and activation with proper voltage/frequency scaling are determined based on Eqns. (14) and (19). As shown in Fig 6, the curve of $IPS_{sum}$ is convex, because there is a trade-off between the reductions in cache misses and clock frequencies as the number of activated cache ways increases. While increasing the number of activated cache ways, $IPS_{sum}$ may increase owing to the reduction in cache misses resulting from more cache capacity allocation. However, it may also reduce core's clock frequency, thus, decrease $IPS_{sum}$ in order to offset the additional power consumed by the more activated cache ways, thus not to violate the temperature limit. In conclusion, there exists a performance-maximal number of activated cache ways, $B_{opt}$ (shown in Fig. 6) at the inflection point where slope of the curve is zero.

## VI. Application of Analytic Formulations at Runtime

In order to apply the analytic formulations explained in Section V to 3-D CMPs at runtime, we need to consider discrete values of the number of assigned and activated cache

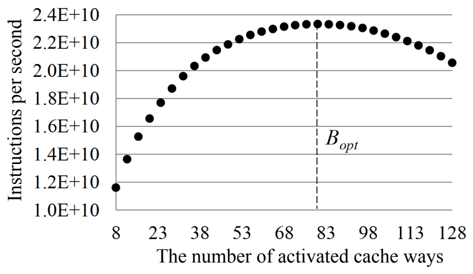

Fig. 6. Results of sum of IPS, $IPS_{sum}$, with respect to the total number of activated cache ways, $B_{active}$.

**Algorithm to find $l_{set}$**

```
1:  for (i = 0; i < M; i++) do
2:      l[i] = 1;
3:  end for
4:  remains = B_active – M;
5:  while (remains > 0) do
6:      grad_min = MAX_VALUE;
7:      for (i = 0; i < M; i++) do
8:          grad = get_grad (i, l[i]);
9:          if (grad < grad_mid) then
10:             grad_mid = grad;
11:             core_index = i;
12:         end if
13:     end for
14:     l[core_index]++;
15:     remains – –;
16: end while
```

Fig. 7. Algorithm to find the number of activated cache ways for each core-set, $l_{set}$

ways, and that of clock frequency of cores with algorithm's computational complexity. Fig 7 shows a greedy algorithm to find the number of activated cache ways for each core-stack, $l_{set}$. This greedy algorithm finds optimal $l_{set}$ based on Eqn. (19). The algorithm starts with initializing the number of activated cache ways for each core-stack as one (line 1-3). The algorithm then finds the core having the smallest gradient, $PL$, and increases the number of activated cache ways in the core-stack by one (line 6-14). This makes the gradient of each core, $PL_i$, almost equal to each other. Lines 6-15 in the algorithm are repeated until the variable *remains* becomes zero. Note that this greedy algorithm can also be applied to find the number of cache ways to be assigned to each core, $b_{set}$, because of the same relation of Eqns. (19) and (14).

Since temperature of a core-stack is determined by power consumed by the core and the cache ways in the same core-stack, the maximum clock frequency that does not violate the temperature limit can be pre-computed for all available range of workload's switching activity (or IPC) and the number of activated cache ways in the core-stack. For fast clock frequency assignment, we prepared a look-up table (LUT) storing the design-time results of (voltage and) clock frequency as shown in Fig 8. Since the clock frequency is not a continuous value in real design, an approximation is required to determine the clock frequency of each core. Thus, if the determined clock frequency based on the LUT does not match any frequency levels of the real system, the nearest one, but not higher one, is selected not to violate the temperature constraint. In our experiment, we used seven steps of voltage/clock frequency levels from 1GHz to 3GHz with 0.333GHz step size.

| IPC | # of activated cache ways | Frequency |
|---|---|---|
| 0.2 ~ 5.0 (0.1 steps) | 1 ~ 32 (1 steps) | 1.0GHz ~ 3.0GHz |

Fig. 8. Look-up table (LUT) storing clock frequency of a core in a core-stack with respect to instructions per cycle (IPC) and the number of activated cache ways in the core-stack.

Since the curve of $IPS_{sum}$ is convex with respect to $B_{active}$ as shown in Fig. 6, a root-finding algorithm such as bisection method [16] can be applied to find the performance-maximal number of activated ways, $B_{opt}$. The bisection method is a general binary search algorithm where the computational complexity is O($log_2^{N_k}$), where $N_k$ is the number of possible ways.

## VII. Experiment

### A. Setups

We performed experiments using 3-D CMPs consisting of a multi-core tier and eight stacked L3 cache tiers. The number of cores varies from four to eight. A core studied in our experiments is based on the architecture of Intel Core 2 Duo *Merom* processor [17], manufactured at 65nm technology node. In each L3 cache tier, four cache ways are directly stacked on a core and each way has capacity of 256KB [18]. To estimate temperature-dependent power consumption for the cores and cache, we used a leakage power model from [12]. To calibrate the power values of core and cache with respect to temperature, we used the product data-sheet [19] and CACTI [18], respectively. The switching power used in this paper is based on the relationship between power, switching activity, frequency, and voltage, which is presented as follows.

$$P_s = C_s \cdot IPC \cdot V_{dd}^2 \cdot f^3 \tag{20}$$

where $C_s$ is the average switching activity and different among applications since the usage of the functional units in microprocessor is different among them. To characterize $C_s$ for each application, we used PTscalar [20], which is a cycle-accurate micro-architecture-level performance and power simulator for Super Scalar architecture.

For 3-D temperature estimation, we employed HotSpot [21] version 5.0 as a grid-based thermal modeling tool. Core floorplan was obtained from [17]. Each core has a size of 7.237mm×5.23mm. The convection capacitance and resistance are 140.4J/K and 0.1K/W, respectively. Other physical parameters such as thickness and thermal conductivity of each tier are shown in Table II. In Table II, *interlayer* is the interface material between two adjacent silicon layers that are connected with TSVs. We modeled *interlayer* as a homogeneous layer with its thermal resistivity and specific heat capacity values. To account for the thermal impact of *interlayer*, we assumed a homogeneous TSVs distribution that allows us to calculate the combined thermal conductance of *interlayer* based on the TSV density. This modeling method has already been justified in [3]. We assumed that each core-stack has a temperature sensor that provides temperature reading at regular intervals

TABLE II. PHYSICAL PARAMETERS FOR THERMAL MODEL

| Layer | Thermal conductance $(W/mK)$ | Heat capacitance $(J/m^3K)$ | Depth $(\mu m)$ |
|---|---|---|---|
| Heat sink | 400.0 | $3.55 \cdot 10^6$ | 6,900 |
| Heat spreader | 400.0 | $3.55 \cdot 10^6$ | 1,000 |
| Thermal interface material | 4.0 | $4.00 \cdot 10^6$ | 20 |
| Core / L3 Cache | 100.0 | $1.75 \cdot 10^6$ | 150 |
| Interlayer | 4.0 | $4.00 \cdot 10^6$ | 20 |

TABLE III. BENCHMARK SUITES

| Suite | Programs |
|---|---|
| HP6 | free_cfd, free_cfd, face_rec, face_rec, sphinx3, sphinx3 |
| LP6 | equake, equake, mcf00, mcf00, mcf06, mcf06 |
| MP6 | free_cfd, face_rec, sphinx3, equake, mcf00, mcf06 |
| HVAR4 | free_cfd, face_rec, equake, mcf00 |
| LVAR4 | face_rec, sphinx3, mcf00, mcf06 |
| MVAR8 | free_cfd, face_rec, face_rec, sphinx3, mcf00, mcf00, mcf06, equake |

(e.g., 200ms). In our experiment, $T_{amb}$ and $T_{max}$ are set to $40^oC$ and $80^oC$, respectively.

Our experiment was performed with SPEC2000/2006 [10], ALPBench [22], and a computational fluid dynamics (CFD) [11] benchmark programs. Among them, we chose six memory-intensive programs and classified the benchmark suites based on the power intensity and variation as listed in Table III. In Table III, HP, MP, and LP indicate high power intensive, middle power intensive, and low power intensive, respectively. The numbers followed by HP, MP, and LP indicate the number of cores used in the experiments. Similarly, HVAR, MVAR, and LVAR indicate high power variation, middle power variation, and low power variation, respectively.

## B. Results

We performed experiments with four different thermal management schemes as follows.
**1) DVFS only**: Clock frequency of each core varies to satisfy the maximum temperature limit, while the whole cache ways are activated.
**2) DCA only**: Each core runs at a fixed clock frequency, i.e., 3GHz, while cache ways are partially activated and assigned to each core in order to maximize system performance and not to violate the temperature limit.
**3) DCA with DVFS** (proposed method): Cache way assignment with voltage/frequency scaling is performed. We assume that the ways farthest from the heat sink are turned off if there are unassigned cache ways.
**4) DCAP with DVFS** (proposed method): Position of activated cache ways is also considered, while applying cache way assignment with voltage/frequency scaling.

Fig. 9 shows the IPS results, which are normalized with respect to that of DCA only. DCA with DVFS improves IPS by 16.22% on average (ranging from 5.4% to 24.2%) compared with DCA only. This results show that allocating more cache capacity to each core while sacrificing clock frequency is effective in achieving further performance improvement under the given temperature constraints. Table IV shows the average clock frequency and the number of activated cache ways per core for DCA only and DCA with DVFS. As shown in Table IV, DCA with DVFS has more activated cache ways with lower clock frequency for all benchmark suites than DCA only. As shown in Fig. 9, DVFS only is the worst scheme because of

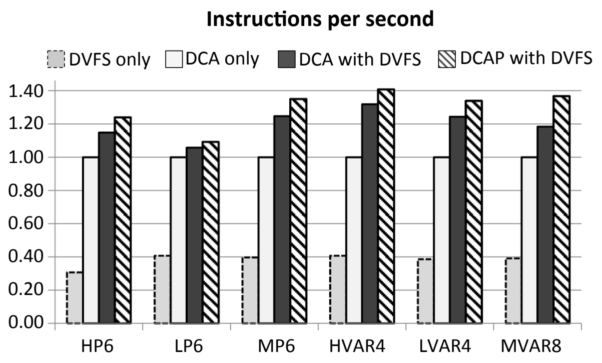

Fig. 9. IPS (instructions per second) results of each benchmark suite.

TABLE IV. AVERAGE CLOCK FREQUENCY, $f_{avg}$ AND AVERAGE NUMBER OF ACTIVATED CACHE WAYS PER CORE, $l_{avg}$.

| Benchmark Suit | DCA only | | DCA with DVFS | |
|---|---|---|---|---|
| | $f_{avg}$ | $l_{avg}$ | $f_{avg}$ | $l_{avg}$ |
| HP6 | 3.00 | 4.75 | 1.55 | 17.10 |
| LP6 | 3.00 | 13.47 | 1.94 | 21.70 |
| MP6 | 3.00 | 5.00 | 1.89 | 18.57 |
| HVAR4 | 3.00 | 4.63 | 1.83 | 17.70 |
| LVAR4 | 3.00 | 4.63 | 2.00 | 16.75 |
| MVAR8 | 3.00 | 4.51 | 2.12 | 18.56 |

the extremely large amount of heat generated from the stacked L3 cache without proper turning cache ways off.

DCAP with DVFS improves IPS by 22.51% on average (ranging from 8.4% to 29.0%) compared with DCA only, and by 7.57% on average (ranging from 3.2% to 13.5%) compared with DCA with DVFS. The difference between DCA with DVFS and DCAP with DVFS, i.e., up to 13.5% performance improvement, shows that mapping the (logically) assigned cache ways to physical cache blocks affects clock frequency of each core and, thus, its instruction throughput. In Fig. 9, the performance of DCAP with DVFS increases as the IPC variations of applications increase. It shows that the effect of more cache resource assignment becomes prominent as the memory-boundedness of applications becomes more significant.

The runtime method used for DCAP with DVFS (explained in Section VI) consists of three methods; finding 1) the number of assigned cache ways to each core, $b_{set}$, 2) the number of activated cache ways in each core-stack, $l_{set}$, and 3) the clock frequency of each core, $f_{set}$. We measured the computational time of each method from the experimental platform, i.e., LG LW 25 laptop with a core running at 3GHz. The computational time is shown in Fig. 10, where the time spent for finding $f_{set}$ is included into the time spent for finding $l_{set}$. Thus, the time spent for finding $l_{set}$ is a little higher than that for $b_{set}$. The time for finding $f_{set}$ is used for looking up the LUT described in Fig. 8. As shown in Fig. 10, the computation time increases with the number of cores. Based on the experimental data in [12], the thermal resistance-capacitance (RC) time constant is usually on the order of hundred milliseconds. The thermal RC time constant also increases with the footprint that is typically proportional to the number of cores. Considering the thermal RC time constant, the runtime overhead shown in Fig. 10 is deemed acceptable. The transition time for voltage (e.g., dc-dc conversion time) and frequency (e.g., phase-locked loop

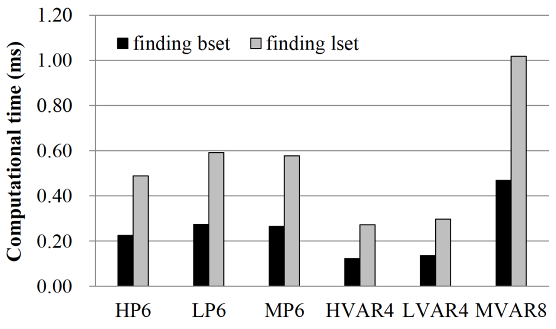

Fig. 10. Computational time of DCAP with DVFS

locking time) add up to $10\mu s$ [23]. When the number of assigned cache ways decreases at runtime, dirty cache blocks in the cache ways that will be turned off must be written back to the off-chip memory for data coherency. When the number of assigned cache ways increases, it costs only wakeup time to switch cache ways back to the active mode from the power-off mode. According to [24], the wakeup time is negligible (i.e., four clock cycles).

## VIII. Conclusion

In this paper, we proposed an integrated solution of cache tuning (i.e., cache resource allocation and turning off unnecessary cache resources) and per-core DVFS for 3-D CMPs with stacked L3 cache. The proposed method employed mathematical formulations to maximize the instruction throughput (i.e., instructions per second) under the given maximum temperature constraint. The experimental results show that the proposed method achieves up to 29% (23% on average) performance improvement compared with the thermal management scheme which performs temperature-aware runtime cache tuning only.

## Acknowledgment

This work is supported by the NanoSys project (ERCAdG-246810) and by the Center for Integrated Smart Sensors funded by the Ministry of Science, ICT & Future Planning as Global Frontier Project" (CISS-2013066998).

## References

[1] B. Black et al., "Die stacking (3D) microarchitecture," In Proc. MICRO, 2006, pp. 469-479.

[2] G. Loh, "3D-stacked memory architectures for multi-core processors," in Proc. ISCA, 2008, pp. 453-464.

[3] A. K. Coskun et al., "Dynamic thermal management in 3D multicore architectures," in Proc. DATE, April 2009, pp. 1410-1415.

[4] C. Zhu et al., "Three-dimensional chip-multiprocessor run-time thermal management," in IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 27, no. 8, pp.1479-1492, Aug. 2008.

[5] K. Kang et al., "Runtime power management of 3-D multi-core architectures under peak power and temperature constraints," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 30, no. 6, pp. 905-918.

[6] G. Sun, H. Yang, and Y. Xie, "Performance/thermal-aware design of 3D-stacked L2 caches for CMPs," ACM Transactions on Design Automation of Electronic Systems, vol. 17, no. 2, pp. 1-20.

[7] M. K. Qureshi and Y. N. Patt, "Utility-based cache partitioning: a low-overhead, high-performance, runtime mechanism to partition shared caches," In Proc. MICRO, 2006, pp. 423-432.

[8] T. Ishihara and F. Fallah, "A non-uniform cache architecture for low power system design," In Proc. ISLPED, 2005, pp. 363-368.

[9] X. Zhou et al., "Thermal-aware task scheduling for 3D multicore processors," IEEE Transactions on Parallel and Distributed Systems, vol. 21, no. 1, Jan. 2010.

[10] *Standard Performance Evaluation Corporation* [Online]. Available: http://www.specbench.org.

[11] *Free Computational Fliud Dynamics* [Online]. Available: http://www.freecfd.com.

[12] W. Liao, L. He, and K. M. Lepak, "Temperature and supply Voltage aware performance and power modeling at microarchitecture level," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 24, no. 7, July 2005.

[13] A. Hartstein et al., "Cache miss behavior: is it $\sqrt{2}$?," In Proc. CF, 2006, pp. 313-321.

[14] J. Kim, S. Yoo, and C.-M. Kyung, "Program phase-aware dynamic voltage scaling under variable computational workload and memory stall environment," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 30, no. 1, Jan. 2011.

[15] H. Everett III, "Generalized Lagrange multiplier method for solving problems of optimum allocation of resources," Operations Research, vol. 11, no. 3, May-Jun. 1963.

[16] G. Corliss, "Which root does the bisection algorithm find?," SIAM Review, vol. 19, no. 2, Apr. 1977.

[17] N. Sakran et al., "The implementation of the 65nm dual-core 64b Merom processor," in Proc. ISSCC, 2007, pp. 106, 107, and 590.

[18] D. Tarjan, S. Thoziyoor, and N. P. Jouppi, "CACTI 4.0," HP Laboratories, Palo Alto, CA, Tech. Rep. HPL-2006-86, Jun. 2006.

[19] Intel, "Intel core2 duo processors and Intel core2 extreme processors for platforms based on mobile Intel 965 express chipset family," Datasheet, 2008, pp. 23-40.

[20] PTscalar [Online]. Available: http://eda.ee.ucla.edu/PTscalar/

[21] W. Huang et al., "HotSpot: a compact thermal modeling methodology for early-stage VLSI design," IEEE Transactions on Very Large Scale Integration Systems, vol. 14, no. 5, May 2006.

[22] M.-L. Li et al., "The ALPBench benchmark suite for complex multimedia applications," In Proc. IISWC, 2005, pp. 34-45.

[23] J. S. Lee, K. Skadron, and S. W. Chung, "Predictive temperature-aware DVFS," IEEE Transactions on Computers, vol. 59, no. 1, Jan. 2010.

[24] H. Homayoun, M. Makhzan, and A. Veidenbaum et al., "Multiple sleep mode leakage control for cache peripheral circuits in embedded processors," in Proc. CASES, 2008, pp. 197-206.

[25] A. C. Nacul and T. Givargis, "Dynamic voltage and cache reconfiguration for low power," in Proc. DATE, Feb. 2004, pp. 1376-1377.

[26] W. Wang and P. Mishra, "Leakage-aware energy minimization using dynamic voltage scaling and cache reconfiguration in real-time systems," in Proc. VLSID, 2010, pp. 357-362.

[27] Z. Jishen, D. Xiangyu, and X. Yuan, "An energy-efficient 3D CMP design with fine-grained voltage scaling," in Proc. DATE, 2011, pp. 1-4.

[28] Kang et al., "Maximizing throughput of temperature-constrained multicore systems with 3D-stacked cache memory," in Proc. ISQED, 2011, pp. 577-582.