Temperature- and Bus Traffic- aware Data Placement in 3D-Stacked Cache

Seunghan Lee, Kyungsu Kang, and Chong-Min Kyung
Department of Electrical Engineering & Computer Science
Division of Electrical Engineering
Korea Advanced Institute of Science and Technology
shlee04@vslab.kaist.ac.kr, kskang@vslab.kaist.ac.kr, kyung@ee.kaist.ac.kr

Abstract— As technology scales, increasing capacity of cache memory leads to increase in leakage power dissipation, especially in three-dimensional (3D) IC with high thermal density. In this paper, we explore how cache data can be mapped on a multi-processor architecture in 3D IC to minimize energy consumption with considering temperature distribution and bus traffic congestion. Simulation results based on ILP (Integer Linear Programming) formulation show that the proposed cache data mapping approach achieves up to 30.7% energy reduction compared to the case of considering temperature distribution only.

I. INTRODUCTION

Three-dimensional (3D) integration is a new promising technology being actively studied to reduce the on-chip interconnect length as a major source of both performance bottleneck and power dissipation in 2D ICs. However, higher power density resulting from multiple die stacking can lead to temperature-related problems such as a reliability (e.g., NBTI) and performance degradation (circuit delay increases in proportion to temperature). Especially leakage power is the most important challenge among temperature-related problems because higher power density leads to increase of leakage power, which in turn leads to increase of chip temperature [1].

This paper focuses on temperature-aware energy minimization, especially for a 3D multi-core system where cache banks are stacked on top of processor cores. A few studies have already investigated the potential of 3D-stacked memory [2-4]. [2] presents 3D-Cacti which estimates latency and energy consumption of 3D-stacked cache according to various cache partitioning parameters (e.g., the number of active device layers available). In [3] and [4], the authors proposed a 3D stacked DRAM architecture to improve memory system performance compared to 2D DRAM organizations. They investigated the potential for using 3D DRAM stacking for both L2 cache and main memory in 3D multi-core architectures. The studies described above showed that 3D-stacked memory increases the bandwidth while reducing the access latency.

There are several temperature-aware cache way placement solutions focusing on 3D-stacked cache for multi-core processors [5-6]. [5] proposed an integer linear programming (ILP)-based processor core/storage block placement solution for both single-core and multi-core embedded designs in order to maximize the system performance (i.e., minimize execution time of workloads) under the specified temperature constraints. [6] proposed a design-time solution of temperature-aware memory mapping to reduce peak temperature of stacked DRAM dies in 3D multi-core architecture. In [7], the author proposed a cache way allocation algorithm which assigns the number of cache ways to each thread according to its characteristics (i.e., the number of cache misses according to the assigned number of cache ways) in order to reduce the power and the peak temperature. In [7], cache ways are allocated to each thread such that increase of cache miss ratio doesn’t exceed the specified threshold. However, this assignment is not optimal solution in terms of energy consumption because it did not consider both cache and processor energy consumption resulting from temperature as well as CPU stall time due to external memory access.

Another key challenge in multi-core systems is bus traffic congestion due to high bandwidth requirement induced from the increasing number of processing cores. There are several works which solve bus traffic-aware cache memory placement problem in order to reduce cache access latency [8-9]. [8] proposed a trace-based bus traffic estimation tool to explore the performance according to different memory allocations. [9] proposed a runtime solution which performs data mapping on NUCA cache at the memory page granularity.

In this paper, we present an ILP-based design-time solution to minimize the system energy consumption (including energy consumption of core, cache, and off-chip memory), especially for a multi-core system where L2 cache is stacked. To the best of our knowledge, this is the first work to minimize the temperature-aware system energy for 3D-stacked L2 cache multi-core architectures through cache way placement. From the proposed method we generate two outputs: (1) number of cache ways to each thread, and (2) the physical location of the allocated cache ways. To do that, we
take into account the temperature distribution (of both core and cache) and bus traffic congestion as well as cache utility (more details in Section 2). In the experimental result, our method yields up to 30.7% (23.8% on average) reduction in energy consumption compared to methods which consider temperature distribution only.

This paper is organized as follows. Section II presents the motivation. Section III presents the problem definition. Section IV and V give a formulation of the allocation problem. Experimental results are given in section VI.

II. MOTIVATION

In this section, we explain the motivation of our work with an example of multi-core systems with 3D-stacked cache. Fig. 1 (a) and (b) shows the example system consisting of two cores (C1 and C2) with two shared L2 cache banks (B1 and B2) stacked on the cores and a crossbar switch connecting the cores (C1 and C2) to the cache banks (B1 and B2). Let us assume that each cache bank consists of eight cache ways. Fig. 1 (c) shows the cache miss rate (i.e., defined as the number of cache misses divided by the number of cache accesses) and the Fig. 1 (d) shows the number of cache accesses per cycle for threads (Qsort and Patricia in MiBench benchmark). In this example, let us assume that Qsort and Patricia are assigned to C1 and C2, respectively. Also, we assume that unused cache ways can be power gated, which consumes only a small leakage power. To evaluate our method, we compared three cache data mapping policies, i.e., conventional method[10], TD-aware, and TD&BCT-aware.

For a thread, the cache miss rate decreases as the number of assigned cache ways increases, shown in Fig. 1 (c), which leads to a reduction of energy consumption of both processor and off-chip memory. However, the increasing number of assigned cache ways induces the increase of cache leakage energy. Thus, we need to find the energy-minimal number of cache ways. A conventional method [10] determines the number of cache ways such that the total energy is minimized with the unused cache way shut down. In Fig. 3 (a), the energy consumption of the conventional method is 0.295J. However, this method, which only focuses on thermal problems in a 2D single-core system, cannot be directly applied to multi-core systems with 3D-stacked cache owing to the significant

![Fig. 1. (a) An motivation example of multi-core systems with 3D-stacked L2 cache (b) A crossbar switch structure connecting cores and cache banks (c) Cache miss rate with respect to the number of allocated cache ways (d) Cache access rate (i.e., the number of cache accesses divided by total number of execution cycles) of Qsort and Patricia.](image1.png)

![Fig. 2. Temperature distribution of each cache bank (B1 and B2)](image2.png)

![Fig. 3. (a) Cache way placement of conventional method (b) TD-aware cache way placement (c) TD&BTC-aware cache way placement](image3.png)

We determine (1) the number of cache ways (i.e., cache capacity) to be used during thread execution and (2) the physical position of the cache ways by taking into account the temperature-distribution of both processing cores and cache banks to reduce the system energy (sum of energy consumption of core, cache, and off-chip memory). To determine the number of cache ways, we consider the temperature-dependent leakage energy consumed in each cache way, which can have different operating temperature, and different leakage energy consumption. In addition, we find the energy-minimal physical position of each cache way, according to its operating temperature. Fig. 3 (b) shows the results of “TD-aware”. In Fig. 3(b), even though the number of allocated cache ways is the same as the conventional method in Fig. 3 (a). In “TD-aware” allocated cache ways are placed at the positions with the lowest operating temperatures. The TD-aware showed the 8.8% energy reduction compared with the conventional method.

In Fig. 3 (b), the solution assumes that there is no bus traffic congestion in the system. However, by exploiting the bus traffic congestion, we obtain further reduction in energy consumption as Fig. 3 (c) shows. In Fig. 3 (c), TD&BTC-aware splits cache data of Patricia between the two cache banks. It is because the number of cache accesses per cycle of Patricia is high as shown in Fig. 1(d). In Fig. 3 (c), TD&BTC-aware reduces the system energy by 14.1% in comparison with the conventional method and 5.8% in comparison with TD-aware.
III. PROBLEM DEFINITION

In our target 3D cache-stacked multi-core architecture, the core layer consists of L processing cores (each of which has its own L1 cache), located next to the heat sink. The cache layers, which form a large shared L2 cache, are composed of SRAM cache banks. The cache layers consist of K cache banks, each of which consists of M cache ways. Assume that the cores access each cache bank through a crossbar switch network [11] and data can be transferred between a core and a cache bank through TSV (Through-silicon via) interconnects. We also assume that the off-chip DRAM (main memory) can be accessed through I/O interfaces. A target 3D multi-core architecture is specified by a set of processors, \( P \), a set of cache banks, \( C_k \), and a set of cache ways, \( W_j \). There is a set of the same number of running threads as processors, \( T_i \). Each core is assumed to be connected to \( K \) cache banks. The cache layers consist of \( C_k \) cache banks and \( W_j \) cache ways allocated to each thread. In Section VII, we will incorporate the temperature and power dissipation of cache blocks and the number of clock cycles spent by the CPU waiting for completion of ideal memory accesses, and the number of clock cycles spent by the CPU waiting owing to bus traffic congestion, respectively, when \( n \) cache ways are allocated and \( n \) cache ways are allocated to the thread \( T_i \). We formulate this problem as follows.

\[
\text{Find } e_{i,n} \text{ and } h_{i,j} \\
\text{such that } E_{\text{system}} = E_{\text{core}} + E_{\text{cache}} + E_{\text{DRAM}} \quad (1)
\]

is minimized.

where \( E_{\text{core}}, E_{\text{cache}}, \) and \( E_{\text{DRAM}} \) are the consumed energy of core, cache, and DRAM, respectively.

subject to

\[
t_{\text{max}} = \max \sum_{i=1}^{M} e_{i,n} \left( X_{i,n}^{\text{comp}} + X_{i,n}^{\text{stall}} + X_{i,n}^{\text{cong}} \right) f \leq t_{d} \quad (2)
\]

where, \( X_{i,n}^{\text{comp}} \) is the number of clock cycles for ideal CPU operation. \( X_{i,n}^{\text{stall}} \) and \( X_{i,n}^{\text{cong}} \) are the number of clock cycles spent by the CPU waiting for completion of ideal memory accesses, and the number of clock cycles spent by the CPU waiting owing to bus traffic congestion, respectively, when \( n \) cache ways are allocated to \( T_i \). \( f \) is the CPU clock frequency.

In the remainder of this paper, we present our solution for the problem as follows. In Section VI, an ILP formulation is given to determine the number of cache ways for each thread, assuming that the allocated cache ways are located on the cache bank which is vertically adjacent to the core running the thread. In Section VII, we will incorporate the temperature distribution and bus traffic congestion into the solution obtained in Section VI to present a complete design-time solution.

IV. DECISION OF THE NUMBER OF CACHE WAYS

The ILP formulation yields the number of cache ways for each thread such that the energy consumption becomes minimal, while the time-to-deadline constraint, \( t_{d} \), is not violated.

**Objective**: Our objective is to minimize the overall system energy, \( E_{\text{system}} \) in Eqn. (1). Since each term in Eqn. (1) depends on the number of allocated cache ways, \( n \), we represent Eqn. (1) as follows.

\[
E_{\text{system}} = \sum_{i=1}^{M} \left( E_{i,n}^{\text{core}} + E_{i,n}^{\text{cache}} + E_{i,n}^{\text{DRAM}} \right) e_{i,n} \quad (3)
\]

where \( E_{i,n}^{\text{core}} \), \( E_{i,n}^{\text{cache}} \) and \( E_{i,n}^{\text{DRAM}} \) are the energy of core, cache, and off-chip memory, respectively, when \( n \) cache ways are allocated to the thread \( T_i \). \( e_{i,n} \) is 1 if \( n \) cache ways are allocated to the thread \( T_i \), and 0 otherwise.

The energy of core, \( E_{i,n}^{\text{core}} \), can be divided into the computational energy and the stall energy and presented as follows.

\[
E_{i,n}^{\text{core}} = X_{i,n}^{\text{comp}} e_{i,n}^{\text{comp}} + X_{i,n}^{\text{stall}} e_{i,n}^{\text{stall}} \quad (4)
\]

where \( e_{i,n}^{\text{comp}} \) and \( e_{i,n}^{\text{stall}} \) are energy per cycle of a core executing the thread, \( T_i \), for computation, and memory stall, respectively. In Eqn. (4), only \( X_{i,n}^{\text{stall}} \) is changed according to \( e_{i,n} \) in Eqn. (3) and can be calculated as follows.

\[
X_{i,n}^{\text{stall}} = n_{i,n}^{\text{DRAM}} \cdot d_{\text{DRAM}} + n_{i,n}^{\text{cache}} \cdot d_{\text{cache}} \quad (5)
\]

where \( n_{i,n}^{\text{DRAM}} \) and \( n_{i,n}^{\text{cache}} \) are the number of accesses to DRAM and cache bank, respectively, and \( d_{\text{DRAM}} \) and \( d_{\text{cache}} \) are the number of cycles for an access to DRAM and cache bank, respectively. The energy of cache, \( E_{i,n}^{\text{cache}} \), is presented as follows.

\[
E_{i,n}^{\text{cache}} = (X_{i,n}^{\text{comp}} + X_{i,n}^{\text{stall}}) e_{i,n}^{\text{cache}_\text{leak}} \quad (6)
\]

where \( e_{i,n}^{\text{cache}_\text{leak}} \) is leakage energy per cycle consumed in \( n \) cache ways in which data of the thread, \( T_i \), is placed. The energy of DRAM, \( E_{i,n}^{\text{DRAM}} \), is presented as follows.

\[
E_{i,n}^{\text{DRAM}} = n_{i,n}^{\text{DRAM}} \cdot e_{i,n}^{\text{DRAM}} \quad (7)
\]

where \( e_{i,n}^{\text{DRAM}} \) is the energy consumption per access to the off-chip DRAM.

**Constraint**:

- **Time-to-deadline**: The maximum execution time among threads, \( t_{\text{max}} \) in Eqn. (2) does not exceed the time-to-deadline \( t_{d} \). In this section, we assume that there is no bus traffic congestion, i.e., \( X_{i,n}^{\text{cong}} = 0 \) in Eqn. (2). Thus, the time-to-deadline constraint is given as follows;

\[
\max \sum_{i=1}^{M} e_{i,n} \left( X_{i,n}^{\text{comp}} + X_{i,n}^{\text{stall}} \right) f \leq t_{d} \quad (8)
\]

- **Total cache capacity**: The total number of cache ways, \( N \), i.e., the sum of all the allocated cache ways to each thread, to be determined in ILP formulation does not exceed the total
number of cache ways in the system. This constraint is given as follows.

\[ N = \sum_{i=1}^{K} \sum_{n=1}^{M} n \cdot e_{i,n} \leq K \cdot M \]  

(9)

V. TEMPERATURE- AND BUS TRAFFIC CONGESTION-AWARE CACHE WAY PLACEMENT

In this section, assuming a pre-determined number of allocated cache ways for each thread, we explain how to determine the physical position of each allocated cache way based on the temperature distribution of each cache bank (explained in Section V.A) and bus traffic congestion (explained in Section V.B).

A. Temperature-aware cache way placement

Given the number of cache ways, \( N \) in Eqn. (9), to be assigned to each thread, we first determine the physical position of each cache way to minimize the sum of temperature-sensitive leakage power dissipation of all the cache ways. As shown in Section II (in Fig. 3), each cache bank \( C_k \) has a temperature distribution owing to the power distribution of each core, which leads to different leakage power consumption among the cache ways. Thus, we need to determine the physical positions of the \( N \) allocated cache ways such that the total sum of leakage power dissipated in each cache way is minimized. To minimize the leakage energy of all the cache ways, we assigned \( N \) cache ways totally to the physical positions in ascending order of their leakage power dissipation in turn. To calculate the leakage power of a cache way, we need to know the temperature distribution of each cache way. In this paper, we used a grid-based temperature model proposed in [12] to obtain the temperature distribution of each core, which leads to different leakage power consumption among the cache ways. Thus, we need to determine the physical positions of the \( N \) allocated cache ways such that the total sum of leakage power dissipated in each cache way is minimized. To minimize the leakage energy of all the cache ways, we assigned \( N \) cache ways totally to the physical positions in ascending order of their leakage power dissipation in turn. To calculate the leakage power of a cache way, we need to know the temperature distribution of each cache way. In this paper, we used a grid-based temperature model proposed in [12] to obtain the temperature distribution of each cache way. To use the grid-based temperature model, we divided each layer into 256 unit blocks (area of each unit block specified in Section V.B). To use the grid-based temperature model, we divided each layer into 256 unit blocks (area of each unit block specified in Section V.B).

B. Bus traffic congestion-aware cache way placement

Cache way placement affects the amount of bus traffic congestion, which will affect the stall time of processor and the processor leakage energy. Thus, we can achieve additional energy reduction by considering the bus traffic congestion. In this section, we assumed that a crossbar switch connects cores and cache banks through its buffers. We also assumed that packets having a fixed size are generated by each core according to Poisson distribution. Given \( S \), the constant service time of the processing element (Cache bank) which is connected to a buffer and \( \lambda_i \), the number of cache accesses per cycle of the thread \( T_i \), the stall time owing to the contention in a buffer shared by \( N \) threads \( T_i \), \( T_j \), ... \( T_N \) can be estimated as follows. [13-14].

\[ G(S, \lambda_{sum}) = \frac{S^2 \cdot \lambda_{sum}} {2(1 - S \cdot \lambda_{sum})} \]  

(11)

where \( \lambda_{sum} = \sum_{i=1}^{N} \lambda_i \).

Eqn. (11) is a nonlinear function with respect to \( \lambda_{sum} \). To exploit Eqn. (11) in our ILP formulation, we approximate this function as linear fitting function, \( G(S, \lambda) = C_1 \lambda + C_2 \). Thus, the number of stall cycle for waiting to access the cache bank \( C_k \) is modeled as follows.

\[ \tau_k = C_1 \left( \sum_{i=1}^{L} \lambda_i \cdot z_{i,k} \right) + C_2 \]  

(12)

where \( \lambda_i \) is the number of cache accesses per cycle of the thread \( T_i \), \( z_{i,k} \) is an binary variable which is 1 if at least one cache way allocated to the thread \( T_i \) is in the cache bank \( C_k \) and 0 otherwise. To exploit the bus traffic congestion, we implement a new ILP formulation by adding Eqn. (12) into the formulation, which is described as follows.

Objective: The objective is to minimize the overall system energy, \( E_{sys} \) in Eqn. (3), by determining the physical position of each cache way. To consider the bus traffic congestion, Eqn. (4) is modified by adding the number of clock cycles spent by CPU stall owing to the bus traffic congestion as follows.

\[ E_{sys} = X_i^{core} \cdot e_i^{core} + (X_i^{stall} + X_i^{cong}) \cdot e_i^{core-stall} \]  

(13)

where \( X_i^{cong} \) is the number of clock cycles spent by CPU stall owing to bus traffic congestion when \( n \) cache ways are allocated to the thread \( T_n \) which is presented as follows.

\[ X_i^{cong} = \sum_{k=1}^{L} n_{i,k}^{cache} \cdot \tau_k \]  

(14)

where \( n_{i,k}^{cache} \) is the number of accesses to the cache bank \( C_k \). In Eqn. (14), \( n_{i,k}^{cache} \) is the product of the number of cache accesses per cache way of the thread \( T_i \) (i.e., \( a_i \)), and the number of cache ways allocated to the thread \( T_i \) which are placed on the cache bank \( C_k \) (i.e., \( v_{i,k} \)) and is presented as follows.

\[ n_{i,k}^{cache} = \sum_{j=1}^{k} (a_{i,j} \cdot v_{i,j}) \]  

(15)

\( a_i \), and \( v_{i,k} \) are presented in Eqn. (16) and (17), respectively.

\[ a_i = \sum_{n=1}^{N} \frac{n_{i,n}^{cache}}{n} \cdot e_i^{cache}, \quad v_{i,k} = \sum_{n=1}^{N} v_{i,n,k} \cdot u \]  

(16)

where \( y_{i,n,k} \) is 1 if the number of cache ways of the thread \( T_i \) in the cache bank \( C_k \) is \( u \) and 0 otherwise.

Eqn. (6) about the cache energy \( E_{cache}^{core} \) is also modified by adding the number of cycles spent by CPU stall and dynamic
cache energy consumption induced from the bus traffic congestion. The modified equation is given as follows.

\[
E_{\text{cache}}^{\text{leak}} = (X_{i}^{\text{comp}} + X_{i}^{\text{stall}} + X_{i}^{\text{cong}}) \cdot \epsilon_{i,n}^{\text{cache, leak}} + \sum_{k=1}^{K} \left( h_{k}^{\text{cache}} \cdot g_{k} \cdot \epsilon_{i,n}^{\text{cache, dyn}} \right)
\]

where \( q_{k} \) is the number of allocated cache ways in the cache bank, \( C_{k} \), and \( \epsilon_{i,n}^{\text{cache, dyn}} \) is the dynamic energy per access to a cache way.

**Constraint:**

- **Time-to-deadline:** The maximum execution time among threads, \( t_{\text{max}} \), in Eqn. (2) does not exceed the time-to-deadline, \( t_{\text{d}} \). Thus, we present a time-to-deadline constraint which considers the bus traffic congestion is given as follows.

\[
\max_{\gamma_{i}} \left( \gamma_{i}^{\text{comp}} + \sum_{n=1}^{W} \gamma_{i,n} \cdot (X_{i}^{\text{stall}} + X_{i}^{\text{cong}}) \right) \cdot f \leq t_{\text{d}} \]  

(18)

- **Limitation to the physical cache way positioning:** Because the physical cache way position is based on the position given from the previous stage, the physical cache way position for cache data of each thread in this stage is limited as follows.

\[
h_{i,j} = 0, \quad T_{i} \in T, w_{j} \in W, W_{j} \notin B\]

(19)

where \( h_{i,j} \) is 1 if the cache way allocated to the thread \( T_{i} \) is the cache way \( W_{j} \), and 0 otherwise.

- **Limitation on the number of allocated cache ways:** The number of determined cache way positions must be equal to the total number of cache ways as follows, where \( e_{i,n} \) is obtained from Eqn. (3).

\[
\sum_{j=1}^{M_{k}} h_{i,j} = \sum_{n=1}^{W} (e_{i,n} \cdot n)
\]

(20)

- **Decision variable equality about the number of cache ways of the thread \( T_{i} \) in cache bank \( C_{k} \)**

\[
\sum_{j=1}^{M_{k}} (h_{i,j} \cdot r_{i,k}) = \sum_{n=1}^{M} (y_{i,k,u} \cdot u)
\]

(21)

where \( r_{i,k} \) is 1 if the cache way, \( W_{j} \), is structurally included in the cache bank, \( C_{k} \), and 0 otherwise. \( y_{i,k,u} \) is 1 if the number of cache ways of the thread, \( T_{i} \), in the cache bank, \( C_{k} \), is \( u \) and 0 otherwise.

- **Decision variable equality about whether the cache way of the thread \( T_{i} \) is placed on the cache bank \( C_{k} \)**

\[
z_{i,k} = \sum_{u=1}^{M} y_{i,k,u}
\]

(22)

where \( z_{i,k} \) is 1 if the cache way allocated to the thread, \( T_{i} \), is in the cache bank, \( C_{k} \), and 0 otherwise.

It should be noted that a non-linear equation can be linearized by Fortet’s linearization method [15] as follows.

\[
\phi_{i,k,u} = n_{i,n} \cdot u \cdot \phi_{i,k,u}
\]

(23)

Other non-linear equations such as Eqn. (14) and Eqn. (17) can also be linearized in a similar way.

**VI. EXPERIMENT SETUP**

We performed experiments using a 3D-stacked cache multi-core system consisting of a core layer (including four cores) and a stacked L2 cache layer (including 4 cache banks). We assumed that a crossbar switch connects cores and cache banks. The base architecture of each core is Alpha 21264 processor. The area of each core is 7.91 mm². We assumed that the area of each cache bank is the same as the size of core. The configuration of each cache bank is given in the Table I [16]. Our method was evaluated with SPEC2000 and Mibench. For evaluation we built two test-sets (denoted by Set A and Set B, respectively) which have different workload characteristics, i.e., the number of execution clock cycles, cache miss rate, cache accesses per cycle. Set A which consists of crafty, gcc, apsi, and vortex in SPEC2000, needs more cache capacity (ways) than those in Set B, which consists of qsort, jpeg, patricia, and dijkstra in Mibench. The number of cache ways needed for cache miss rate less than some specified value is larger in Set A than Set B.

**TABLE I. L2 CACHE BANK CONFIGURATION**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cache bank size</td>
<td>512 KB</td>
</tr>
<tr>
<td>Line size</td>
<td>64 B</td>
</tr>
<tr>
<td>Associativity</td>
<td>8</td>
</tr>
<tr>
<td>Technology</td>
<td>32 nm</td>
</tr>
<tr>
<td>Area</td>
<td>7.91 mm²</td>
</tr>
</tbody>
</table>

**TABLE II. PHYSICAL PARAMETERS OF 3D STRUCTURE**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core/Core layer thickness</td>
<td>150 um</td>
</tr>
<tr>
<td>Core/Core thermal conductivity</td>
<td>100 W/mK</td>
</tr>
<tr>
<td>Heat sink thickness</td>
<td>6900 um</td>
</tr>
<tr>
<td>Heat sink thermal conductivity</td>
<td>400 W/mK</td>
</tr>
<tr>
<td>Spreader thickness</td>
<td>1000 um</td>
</tr>
<tr>
<td>Spreader thermal conductivity</td>
<td>400 W/mK</td>
</tr>
<tr>
<td>TIM thickness</td>
<td>20 um</td>
</tr>
<tr>
<td>TIM thermal conductivity</td>
<td>4 W/mK</td>
</tr>
<tr>
<td>Ambient temperature</td>
<td>45°C</td>
</tr>
</tbody>
</table>
We obtained $e_{i,j}^{core,comp}$ and $e_{i,j}^{core,stall}$, core energy for each thread in the ILP formulation, i.e., from PTScalar tool [1]. We also obtained energy of bank drain and DRAM for each thread, used in the ILP formulation, i.e., $e_{cache,dram}^{i,j}$, $e_{cache,leak}^{i,j}$, and $e_{DRAM}^{i,j}$, from Cacti 6.0. We obtained the workload characteristics such as the number of execution clock cycles, cache miss rate, and the number of cache accesses per cycle from SimpleScalar 3.0. For temperature calculation, we used HotSpot [12] version 3.0.2. Physical parameters for temperature estimation such as layer thickness, material thermal resistivity and ambient temperature are shown in Table II.

VII. EXPERIMENT RESULT

We performed experiments with three different cache data mapping policies, i.e., TD-unaware, TD-aware, and TD&BTC-aware (proposed) for the two test sets (Set A and Set B). Each policy has a different assumption to solve the cache data mapping problem as follows:

1. TD-unaware: Assume that the temperature distribution of the whole cache layer is uniform. This assumption can be considered as an extension of 2D ICs to determine the temperature of each cache bank. [10]

2. TD-aware: Assume that the temperature is different for each cache bank and there is no bus traffic congestion.

3. TD&BTC-aware (proposed): Assume that temperature is different for each cache bank and there is bus traffic congestion.

Fig. 4 shows the result of energy consumption for TD-unaware, TD-aware, and TD&BTC-aware. TD&BTC-aware yields 17.5% and 30.0% in energy reduction compared to TD-unaware for Set A and Set B, respectively. This is due to the consideration of the temperature distribution and bus traffic congestion together in the proposed. It was observed that, in Set B, the cache miss rate of each thread converges at smaller number of assigned cache ways than in Set A. Thus, each thread in Set B is assigned the smaller number of cache ways than those in Set A. (In our experiment, the sum of the number of cache ways assigned to the threads in Set A and Set B were 26 and 16, respectively.) The smaller is the number of cache ways, the more chances to place each cache way to exploit both temperature distribution in cache layer and bus traffic congestion. Thus, in the evaluation of our method (TD&BTC-aware), Set B yields more energy reduction than Set A as shown in Fig. 4.

Fig. 4 shows that while the energy consumption of cache in TD-aware is lower than that in TD-unaware, the energy consumption of core in TD-aware is higher than that in TD-unaware. In total, this leads to the larger total energy consumption for Set B. For the energy consumption of cache, TD-aware exploits the temperature distribution of the cache layer, which leads to the decrease of the temperature-sensitive cache energy. For the energy consumption of core, TD-aware can lead to an increase of stall cycles in cores due to unintelligent cache way placement by not considering the bus traffic congestion. In Fig. 4 (b), even though cache energy consumption of TD-aware is reduced by 11% compared to TD-unaware, core energy consumption of TD-aware increases by 15% which is due to the bus traffic-caused stall compared to TD-unaware, which leads to the increase of total energy consumption (1.0% increase in energy consumption). In summary, cache data mapping needs to consider the bus traffic congestion as well as the temperature distribution.

VIII. CONCLUSION

In this paper, we address the problem of cache data mapping for a multi-core architecture with 3D-stacked L2 cache to minimize the overall system energy. Unlike the classical approaches, our method considers bus traffic congestion as well as temperature distribution. It places cache data to the specific physical position by exploiting the trade-off between the energy induced by bus traffic congestion and the energy induced from temperature distribution of the cache layers. We solved this problem by formulating ILP model and the experiment results show that our method yields up to 30.0% improvement in energy reduction compared to an existing method [10].

ACKNOWLEDGMENT

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MEST) (No.2010-0000823) and IT R&D program of MKE/KEIT [ K1002134 , Wafer Level 3D IC Design and Integration].

REFERENCES