Thermal-Aware Job Allocation and Scheduling for Three Dimensional Chip Multiprocessor

Shaobo Liu, Jingyi Zhang, Qing Wu, and Qinru Qiu
Department of Electrical and Computer Engineering
Binghamton University, State University of New York
Binghamton, New York 13902, USA
{sliu5, jzhang5, qwu, qqiu}@binghamton.edu

Abstract - *In this paper, we propose a thermal-aware job allocation and scheduling algorithm for three-dimensional (3D) chip multiprocessor (CMP). The proposed algorithm assigns hot jobs to the cores close to the heat sink and cool jobs to the cores far from the heat sink, subject to thermal constraints. The direct effect of the proposed algorithm on a 3D-CMP system is that, the heat from hot jobs is removed off the chip faster than temperature-aware methods. Therefore we are able to keep the chip cooler and in better thermal condition. Experimental results show that, comparing to the temperature-aware method, our algorithm achieves: 1) less hot spots; 2) better performance; 3) smaller temporal temperature variation; 4) lower peak temperature. The proposed algorithm reduces hot spots by more than 95% when workload contains cool jobs; and by 36% when workload does not contain cool jobs. It also boosts the system performance by 5% on average under various workloads. The temporal temperature variation is reduced by 60% and its standard deviation is decreased by 50%. In addition, the proposed algorithm achieves 1.8^oC ~5^oC reduction in peak temperature.*

Keywords

Scheduling, thermal management, three dimensional, chip multiprocessor, job allocation

I. Introduction

The design paradigm of processor architecture has been recently evolving from uni-core, multi-core to many-core. It is reported that up to hundreds of cores will be integrated on a *chip multi-processor* (CMP) in the years to come [1]. In order to integrate more functionality and achieve high performance, both low communication latency and high integration density are critical to the CMP architecture. However, the traditional two dimensional (2D) planar COMS fabrication process shows poor scalability in interconnect/wiring [2] and it faces challenge in implementing CMP. Three dimensional (3D) planar CMOS fabrication technology is recently proposed as a promising approach to solve that problem. The 3D CMOS fabrication technology vertically stacks two or more active silicon layers together through inter-die vias, and hence obtains high transistor integration density. 3D CMP architecture also improves the communication latency significantly by reducing the global and semi-global wiring length [3-5].

However, the high transistor density leads to the high power density; moreover, the overall power dissipation almost scales up linearly with the number of cores. The resultant on-chip temperature rise not only decreases reliability, and degrades performance, but also super-linearly increases packaging & cooling cost [6]. The cooling cost increase directly leads to the increase of the overall cost because the packaging and cooling cost amount to a considerable portion of the total cost of computing systems [7]. The elevated on-chip temperature also reduces the lifetime and accelerates the permanent mechanical failures of the chip [8]. The lifetime of the chip decreases by half with a 10~15^oC increase of operation temperature [8]. The leakage current contributes a significant portion to total power consumption beyond 45nm process node [9, 10]; large leakage power results in a higher power dissipation and higher on-chip temperature. The increased on-chip temperature will in turn results in the exponential increase of leakage power [11]. The positive feedback between leakage current and temperature can easily cause the thermal runaway without proper thermal management. The high operation temperature also decreases the carrier mobility and degrades the performance of the circuit.

Another issue in 3D chip design is the temperature fluctuation across the chip due to disparate heat transfer efficiencies at different layers. The large spatial temperature fluctuation results in degrading system reliability [12]. Hence it is important to regulate the on-chip temperature so as 1) to keep 3D CMP chip operating at the moderate temperature range; 2) to reduce the temperature fluctuation across the chip.

The on-chip temperature is traditionally managed by advance packaging and cooling solutions. As the power density continues to double every 18 months, such solutions for temperature management are becoming prohibitively expensive [7]. In the past years researchers have been actively exploring alternative techniques to control on-chip temperature [13-15]. The majority of those techniques are reactive. When the on-chip temperature is over some predefined threshold, a proper technique is applied to reduce the temperature or to keep the temperature below the critical temperature of the chip. Such techniques include dynamic power management (DPM), dynamic voltage and frequency selection (DVFS), dynamic thermal management (DTM), fetching throttling and task migrations, etc. Those techniques are designed specifically for managing the thermal behavior of 2D chips.

The ultimate goal of thermal management is to guarantee that the operating temperature of the chip is not over its critical temperature such that the stability, reliability and performance are achieved. 3D integration complicates heat transfer and new thermal management approaches for 3D CMP architecture need to be explored. The on-chip temperature is inherently controlled by the heat staying on the chip. If the heat could transfer off chip instantly, then the on-chip temperature would not go up at all. Similarly, if more heat could transfer off chip at a given time interval, the on-chip temperature will be lower. From this viewpoint, we propose a novel thermal aware job allocation algorithm in this paper. The proposed algorithm always assigns power-intensive jobs to the core where the heat could be removed off chip most quickly subject to the thermal constraints.

Based on the proposed algorithm, power-intensive jobs are assigned to the cores sitting right behind the heat sink. So the majority of heat transfers to the ambient going through a very short heat conduction path. One of the direct benefits is that few cores are passively heated up due to the heat conduction.

The highlight of the proposed algorithm can be summarized as follows:

1) Minimize the time of heat staying on the chip and the resultant adverse impact.
2) Avoid detrimentally heating up the cores located on the heat conduction path.
3) Reduce the temporal temperature fluctuation and improve the reliability.
4) Maximize the performance by reducing the thermal management event.

The rest of the paper is organized as follows. Section II introduces the related work. The thermal model is presented in Section III. Thermal-aware job assignment and scheduling algorithm is explained in Section IV. The experimental results and summaries are presented in Sections V and VI, respectively.

II. Related Work

The high integration density in 3D CMP complicates the thermal modeling and management; and the existing thermal management techniques for 2D CMP cannot be directly applied to 3D CMP. Therefore, new approaches which focus on 3D CMP thermal management and thermal optimization should be explored.

There are several techniques proposed targeting at design stage optimization for 3D chips. A thermal-aware floorplanning algorithm is proposed in [16]. Authors in [3] further propose an algorithm, which also aims at thermal-aware floorplanning, but takes into account the interconnect power consumption. A thermal-aware placement algorithm is proposed in [17] for 3D chips.

A "thermal herding" approach is proposed in [18] for 3D chip in order to improve the efficiency of heat removal. That approach puts the individual function units across the multiple active silicon layers. The parts with high switch activity are placed on the layer near to the heat sink; vice versa.

A job allocation, scheduling and voltage & frequency selection algorithm is proposed in [19]. The goal of that algorithm is to reduce the power consumption subject to the thermal constraints. Another thermal management algorithm for 3D CMP is proposed in [20], where some policies based on job migration and DVFS are discussed and evaluated.

Most recently, two temperature aware job assignment algorithms are present in [21, 22]. From the viewpoint of statistics, Adapt3D approach [21] always assigns the upcoming job to the coolest core to achieve thermal balance across the chip. It is a temperature-aware scheduling policy in essence. Authors in [22] claim that the temperature of vertically aligned-cores has very strong correlation. Base on that, the vertically aligned cores are wrapped up into super cores. Accordingly, tasks are also wrapped into super tasks. Then the hottest super job is assigned to the coolest super core in order to achieve the thermal balance. In both algorithms, however, the efficiency that the core removes the heat off the chip is not considered. If the coolest core is farthest from the heat sink, the heat of that core is transferred off chip most inefficiently. In terms of those two policies, the hottest job will be assigned to that coolest core; the new hot spots will be easily generated; moreover, the heat staying longer on chip will detrimentally heat up neighboring cores too.

As opposite to the above two algorithms, in this paper we propose a thermal-aware job allocation algorithm, which assigns jobs to cores from where heat is always removed off most quickly and efficiently subject to the thermal constraints. Since the heat is removed off chip more quickly, our algorithm reduces the hot spots, mitigates temporal temperature variation, decreases the peak temperature and also boosts the system performance, as shown in the experimental results.

III. Thermal Model and Assumptions

In this section, we are going to introduce the thermal model of a single core, the thermal model of 3D CMP, and the thermal profile of a single job. We start with introducing the thermal model of a single core, since it is the simplest one.

A. Thermal Model of Single Core

There exists a duality between the first order electrical RC model and the spatial one-dimensional thermal model. We take a lumped RC model as the thermal model of a single core [23], as shown in Figure 1. The power dissipation is equivalent to the current source, and the ambient temperature equivalent to voltage source.

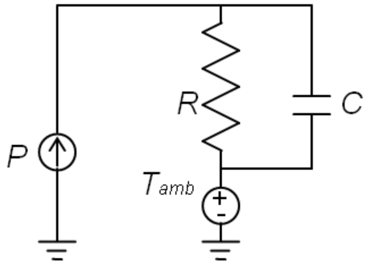


Figure 1 A lumped RC thermal model.

Assume the ambient temperature is T_{amb}, and the initial temperature of the core is T_{init}, the average power consumption of the core is *P* at the given time, then we have:

$$C\frac{d(T(t)-T_{amb})}{dt} = P - \frac{T(t)-T_{amb}}{R} \quad (1)$$

By solving the above differential equation, we get the core temperature at time *t*

$$T(t) = P\cdot R + T_{amb} - (P\cdot R + T_{amb} - T_{init})e^{-t/RC} \quad (2)$$

where *R* and *C* represent the equivalent thermal resistance and the equivalent thermal capacitance from the core to the ambient, respectively.

B.Thermal Profile of Single Job

Once a job is in the job queue of the scheduler, we assume that parameters of the job such as the worst case execution time and the average power dissipation are given. Consider a job J_i with average power dissipation P_i and worst case execution time w_i. If thermal model of the core is given by equation (2); then we can calculate the temperature of the core after J_i is executed,

$$T_i(w_i) = P_i \times R + T_{amb} - (P_i \times R + T_{amb} - T_{init})e^{-w_i/RC} \quad (3)$$

where T_{init} is the temperature of the core right before it executes J_i.

The temperature change of the core after the execution of job J_i is

$$\Delta T_i = T_i(w_i) - T_{init} \quad (4)$$

Plugging equation (2) into equation (3), we have

$$\Delta T_i = (P_i \times R + T_{amb} - T_{init})(1 - e^{-w_i/RC}) \quad (5)$$

Assume the processor executes an infinite sequence of the same job J_i, then we have

$$T_i(\infty) = P_i \times R + T_{amb} - (P_i \times R + T_{amb} - T_{init})e^{-\infty/RC} \quad (6)$$

The above equation is reduced to

$$T_{i,steady} = T_i(\infty) = P_i \times R + T_{amb} \quad (7)$$

where $T_{i,steady}$ is the steady state temperature of job J_i.

Observing equation (7), we find that the smaller equivalent thermal resistance, the lower steady state temperature of the core. Plugging equation (7) into equation (5), we have:

$$\Delta T_i = (T_{i,steady} - T_{init})(1 - e^{-w_i/RC}) \quad (8)$$

$e^{-w_i/RC}$ is always less than 1; hence the temperature change direction is determined by $T_{i,steady} - T_{init}$. If $T_{i,steady}$ is larger than T_{init}, then the temperature goes up; otherwise, goes down.

C. Thermal Model of 3D Chip Multiprocessor

Three-dimensional (3D) chip multiprocessor (CMP) contains multiple active silicon layers, as shown in Figure 2. Each active silicon layer contains processing units, which are also called cores. As we can see from Figure 2, the thermal interface material (TIM) is included in CMP package in order to improve heat transfer efficiency. The carrier layer is attached to one side of the 3D CMP so that the chip could be easily soldered on the printed circuit board (PCB); the heat sink to the other side helps remove the heat off the chip more quickly.

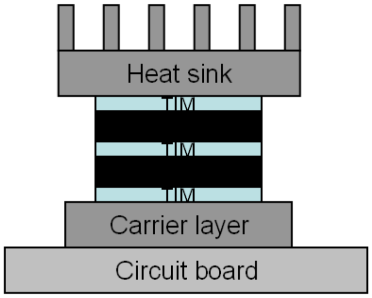


Figure 2 Cross sectional view of a 3D chip.

For the simplicity of thermal modeling, we ignore the horizontal lateral heat conduction and thermal interaction between any two cores. The distance from different cores to the heat sinks varies; hence the thermal resistances of cores are disparate. Considering a core located in silicon layer *j*, its equivalent thermal resistance to the ambient R_j can be calculated as,

$$R_j = \sum_{k=1}^{j} R_{k,k-1} + R_{0,hsk} + R_{hsk,amb} \quad (9)$$

where $R_{k,\ k\text{-}1}$ represents the thermal resistance of between silicon layer *k* and *k*-1. $R_{0,hsk}$ represents the thermal resistance between silicon layer 0 and heat sink. $R_{hsk,amb}$ the thermal resistance from the heat sink to the ambient. Then based on equations (1) and (2), we know the thermal behaviors of the given core *j*. From equation (2), we know the thermal resistance plays a key role in the course of heat transferring off chip. For the given power dissipation, smaller thermal resistance will result in lower on-chip temperature. From the viewpoint of dynamic thermal management, the lower on-chip temperature is desired. Therefore it is important that job should be always assigned to cores with lower thermal resistance subject to the thermal constraints.

IV. Scheduling Algorithms

The majority of heat is transferred off chip through conduction; hence the thermal resistance is a decisive factor on how quickly the heat could be removed off chip. The chip temperature is inherently decided by the amount of heat staying on chip. The more the heat spreads to the ambient in a given time interval, the cooler the chip. In another word, the chip will keep cooler if the heat is always removed off the chip more quickly. Base on that, we propose a job assignment algorithm taking into account the thermal resistance of cores. The basic idea of our algorithm is that hot jobs are assigned to cores with lower thermal resistance; and cool jobs assigned to cores with high thermal resistance. Heat from hot jobs dominates the temperature of the chip. Base on the proposed algorithm, the heat from hot jobs is easily removed and there should be fewer hot spots generated on chip.

A.Thermal-Aware Scheduling

As shown in Section III.C, the core closer to the heat sink has smaller thermal resistance and the core farther away has larger one. With the same workload, cores having smaller thermal resistance will get the lower steady temperature, which has been proven by equation (7). Therefore, in order to achieve thermal balance across the chip, it is of great importance to assign hot tasks to cores with smaller thermal resistance and vice versa.

Let set $\mathcal{C} = \{C_m|m=1,2,\ldots,M\}$ represent all cores of a 3D chip, where M is the total number of cores. Without loss of generality, assume C_m has smaller thermal resistance than C_{m+1}; C_1 has the smallest thermal resistance and C_M has the largest one. Let set $\mathcal{J} = \{J_n|n=1,2,\ldots,N\}$ represent jobs in the scheduler waiting for the assignment, where N is the total number of jobs. The jobs in $\mathcal{J}$ are dispatched to the cores by the scheduler upon the scheduling interval. In order to simplify the discussion, let M equal N first. Later on, this constraint will be relaxed.

Before dispatching any job to the core, all jobs in $\mathcal{J}$ are sorted in the descending order based on their power dissipation and we get a sorted job sequence J_{n_1}, J_{n_2}, …, J_{n_N}, where $n_k \in \{1,2,\cdots,N\}$. Then the first job J_{n_1} in $\mathcal{J}$ is tentatively assigned to C_1. We need to predict the temperature $T_{n_1,C_1}(w_{n_1})$ of core C_1 after it finishes job J_{n_1} based on equation (3);

$$T_{n_1,C_1}(w_{n_1}) = P_{n_1} \times R_{C_1} + T_{amb} - (P_{n_1} \times R_{C_1} + T_{amb} - T_{init,C_1})e^{-w_{n_1}/R_{C_1}C_{C_1}} \qquad (10)$$

where R_{C_1}, C_{C_1} are the thermal resistance and capacitance of C_1, respectively; and T_{init,C_1} is the initial temperature of C_1 before running job J_{n_1}. It's important to point out that, T_{init,C_1} is the actual core temperature that is monitored by on-chip temperature sensors. Therefore a temperature reading must be performed before the prediction using equation (10).

It is also important to point out that, equation (10) is a temperature predictor which estimates the temperature of a given core after a specific job is executed. In order to reduce the computation overhead, the lateral thermal interactions among adjacent cores are not considered in (10). We don't need high accuracy at this stage because the temperature predictor is used as the temperature threshold check in our algorithm. But in our experiments, we use Hotspot4.1 [23] with an accurate thermal model that includes both lateral and vertical thermal interactions among different cores.

If the predicted temperature $T_{n_1,C_1}(w_{n_1})$ is no more than the chip critical temperature $T_{critical}$, then the assignment of job J_{n1} to core C_1 is finalized. However, if $T_{n_1,C_1}(w_{n_1})$ is over the chip critical temperature, its indication is interpreted as one of the followings,

1) The temperature of core C_1 has already been high, and the assignment of new job J_{n1} to it will generate new hot spot on core C_1.
2) The power dissipation of job J_{n1} is too high for the current thermal situation of core C_1, and we should assign a job with lower power dissipation to core C_1.

In either case, job J_{n1} cannot be assigned to core C_1 in order to avoid the new hot spots to generate. Instead, the scheduler tentatively assigns job J_{n1} to core C_2 at next place. If the assignment of job J_{n1} to core C_2 can be not finalized either, then the scheduler tentatively assigns job $J_{n1}$to core C_3 and so on. Generally if a job is waiting for allocation, then the scheduler always tries to assign the job to the cores who have not been assigned any job before in the given order of the core's thermal resistance until the assignment is finalized. Say job J_{n1} is finally assigned to core C_k, then cores indexed from 1 to k-1 have been tentatively assigned job J_{n1} to by the scheduler; however, none of tentative assignments could be finalized due to the thermal constraints.

If job J_{n1} cannot be assigned to any core due to the thermal constraints, then the allocation of job J_{n1} is delayed to the next scheduling interval. In the similar way, the scheduler assigns jobs J_{n_2}, …, J_{n_N}. The thermal aware job assignment algorithm is presented in Algorithm 1. The input of Algorithm 1 is *sort_direction*, which has two possible values: *ascendingly* and *descendingly*. Initially *sort_direction* has value *ascendlingly*. Later on we will explain how to obtain the value of *sort_direction*.

Algorithm 1 Thermal aware job assignment

```
Input: sort_direction
1.  sort all cores in set $\mathcal{C}$ ascendingly based on thermal
    resistance, and get sequence $C_1$, $C_2$, …, $C_M$
2.  sort all jobs in set $\mathcal{J}$ <sort_direction> in terms of power
    dissipation, and get job sequence $J_{n1}$, $J_{n2}$, …, $J_{nN}$
3.  while $\mathcal{J}$ not empty do
4.      for $m$ = 1:$M$ do
5.          if $C_m$ is not assigned any job do
6.              tentatively assign the first job $J_{nk}$ in $\mathcal{J}$ to $C_m$
7.              calculate $T_{nk,Cm}(w_{nk})$
8.              if $T_{nk,Cm}(w_{nk}) \leq T_{critical}$ do
9.                  officially assign job $J_{nk}$ to $C_m$
10.                 remove $J_{nk}$ from $\mathcal{J}$
11.                 break;
12.             end if
13.         end if
14.         if $m == M$ do
15.             remove $J_{nk}$ from $\mathcal{J}$    // $J_{nk}$ cannot be assigned
                                                     // to any core
16.         end if
17.     end for
18. end while
```

Line 15 of Algorithm 1 indicates that job J_{n_k} is removed from job set $\mathcal{J}$ since it cannot be assigned to any core. Job J_{n_k} will be put back into job set $\mathcal{J}$ at the next scheduling interval for assignment.

As long as the thermal constraints are not violated, Algorithm 1 always assigns the hottest job to the core with smallest thermal resistance, the second hottest job to the core with second smallest thermal resistance, and so on. In the simplest case, job J_{n_k} is assigned to core C_k, where $1 \leq k \leq N$. However, when mismatch happens between J_{n_k} and C_k due to the thermal constraints, the scheduler then assigns job J_{n_k} to the core which has not been assigned any job and also has the smallest thermal resistance among C_k, C_{k+1}, C_M subject to the thermal constraints. Based on this job assignment policy, we guarantee that hot jobs are

Algorithm 2 get_sort_direction()

```
1.  calculate the average temperature $T_{s,\,avg}$ of $B$ cores with
    smallest thermal resistance
2.  calculate the average temperature $T_{l,agv}$ of $B$ cores with largest
    thermal resistance
3.  $\Delta T_{S,L} = T_{l,agv} - T_{s,\,avg}$
4.  if $\Delta T_{S,L} > T_{ub}$ do        // temperature distribution unbalanced
5.      sort_direction = ascendingly
6.  else if $\Delta T_{S,L} < 0$ do
7.      sort_direction = decendingly
8.  else do
9.      keep the value of sort_direction
10. end if
11. return sort_direction
```

assigned to the cores with small thermal resistance and the cool jobs to the cores with large thermal resistance. As a consequence, the large amount of heat from hot jobs can be easily transferred off chip; and the small amount of heat from cool jobs will not elevate the on-chip temperature too much. The complexity of Algorithm 1 is $O(M^2)$ in worst case and $O(M)$ in best case.

As discussed before, some jobs cannot be assigned to any core due to the thermal constraints. Accordingly, some cores will not get any job. From the algorithm 1, we know that the reason the core cannot be assigned with any job is that the temperature of the core is already high. Even the job with lowest power dissipation will cause temperature over the critical temperature of the chip. Hence it is necessary that the hot core be idle for one scheduling interval to cool itself down and reduce its temperature.

If *sort_direction* always has value *ascendingly* in Algorithm 1, then hot jobs are always first assigned to the cores with small thermal resistance; accordingly, the cores with small thermal resistance may become very hot, which can result in the unbalanced temperature distribution across the chip.

Before we introduce the solution to the issue of unbalanced temperature distribution, several notations are needed in order to simplify the discussions.

T_{ub} the critical point indicating whether or not the spatial temperature distribution is balanced

B the number of cores used for evaluating if the temperature distribution is balanced

$T_{s,\,avg}$ the average temperature of B cores with smallest thermal resistance

$T_{l,agv}$ the average temperature of B cores with largest thermal resistance

$\Delta T_{S,L}$ the different between $T_{l,agv}$ and $T_{s,\,avg}$.

If the temperature difference $\Delta T_{S,L}$ between $T_{l,agv}$ and $T_{s,\,avg}$ is over T_{ub} , then the temperature distribution is regarded as unbalanced. In order to reduce the spatial temperature variation, the scheduler will dispatch jobs with low power dissipation to the core with small thermal resistance until $\Delta T_{S,L}$ decreases to zero. In others words, *sort_direction* in Algorithm 1 should be *ascendingly* before the scheduler dispatches jobs to the cores, which is determined by the line 5 of Algorithm 2. Note that we calculate the average temperature on B cores so that $\Delta T_{S,L}$ excludes the disturbance to some degree and reflects the unbalanced temperature distribution across the chip more accurately. B should be set to a proper value based on the floorplanning. In our experiments, B is set to 4.

Algorithm 2 shows how to obtain the value of *sort_direction*, which is needed in Algorithm 1 when the scheduler dispatches jobs. Putting Algorithm 1 and Algorithm 2 together, we get the proposed job allocation and scheduling algorithm, as presented in Algorithm 3, which is self-explanatory.

Algorithm 3 the proposed job allocation algorithm

```
1. initially set sort_direction to decendingly
2. while scheduler dispatches jobs do
3.     run Algorithm 2 to get the value of sort_direction
4.     run Algorithm 1 to allocate jobs to the different cores
5. end while
```

Note that when deriving Algorithm 1, we assume the number of jobs N is equal to the number of M. If N is fewer than M, the jobs can be scheduled based on Algorithm 1 without any problem. However, if N is larger than M, then Algorithm 1 cannot be directly. In that case, N jobs should be grouped into M super jobs.

First we need to sort N jobs; and then group the first M jobs into first super job, the second M jobs into the second super job, and the $\lfloor N/M \rfloor$th M jobs into the super job, where symbol "$\lfloor x \rfloor$" indicates the largest integer which is no more than x. If N is not the multiple of M, let $M = \lfloor N/M \rfloor \times M + r$, where r is an integer ranging from 1 to M-1. There are r jobs left after we get $\lfloor N/M \rfloor$super jobs. Put the first left job into the first super job, the second left job into the second super job, …, and the rth left job into the rth super job. After that, we will obtain M super jobs, and Algorithm 1 can be used to schedule each super job.

Finally, we need to point out that DVFS policy can be easily integrated into the proposed algorithm to further reduce the peak temperature of the chip and save energy if needed.

In summary, we have proposed a thermal aware job assignment algorithm for 3D CMP. The framework of the proposed algorithm consists of two steps:

1) Assign hot job to the core with small thermal resistance and cool job to the core with large thermal resistance, as presented in Algorithm 1.
2) If the temperature distribution across the 3D chip is unbalanced, the jobs with low power dissipation are assigned to the cores with small thermal resistance so that the spatial temperature variation could be reduced.

V. Experimental Results

In this section, we will first introduce the experiment setup and the benchmarks used in our experiments. Then the simulation results for the proposed scheduling algorithm are presented.

We have developed a discrete event-driven simulator in C/C++ and implement the proposed algorithm through integrating Hotspot4.1 [23] into our simulator. The lateral and vertical thermal interactions among adjacent cores are all accurately modeled in our experiments when we use Hotspot4.1 to emulate the thermal behavior of different cores. As comparison, a temperature-aware job scheduling algorithm is implemented as the benchmark algorithm. This temperature-aware algorithm is similar to the one in [21] except that we use one temperature for the whole core. We adopt the similar floorplans in [20, 22] which do not separate the cores from L2 cache. For a given core, we assume that there is a sensor for the hot-spot temperature, instead of the average temperature across the core.

All experiments are conducted on Linux workstation equipped with an Intel Xeon X5472 processor and 4GB of RAM. The proposed algorithm requires less than 60 seconds of CPU time for each benchmark used in our experiments.

A.Experimental Setup

In this subsection we will introduce the experimental setup and the benchmark used in the experiments.

We choose a two-layer 16-core 3D CMP architecture. The floorplan similar to the one in [20, 22] is adopted, where L2 cache is not separated from the core. Each active silicon layer contains 8 Alpha microprocessor-like cores [24]. Cores in the top layer are indexed from 1 to 8; cores in the bottom layer indexed from 9 to 16. Each core has a size of 4mm×8mm. The thickness of the top layer silicon is 50μm; the thickness of the bottom layer 500μm. As shown in Figure 2, the bottom silicon layer needs provide the support to mount the chip on the PCB board; and it is 10X times thicker than the top silicon layer. The bottom silicon layer is closer to the copper heat sink. There exists a thermal interface material layer between two active silicon layers. Other parameters used in our experiments are listed in Table 1 [23].

The critical temperature of the chip is set to 85 ^{o}C. The scheduler dispatch jobs to the core every 8ms; accordingly the temperature of each core is read once by the scheduler every 8ms so that the scheduler has the latest temperature information every time it schedules jobs.

Table 1. Thermal model parameters for Hotspot 4.1.

Parameter	Value
Thermal conductivity (silicon)	100 $W/(m \cdot K)$
Thermal conductivity (copper)	400 $W/(m \cdot K)$
Thermal conductivity(TIM)	4 $W/(m \cdot K)$
Thermal capacitance per unit volume(silicon)	1.75×10^6 $J/(m^3 \cdot K)$
Thermal capacitance per unit volume(copper)	3.55×10^6 $J/(m^3 \cdot K)$
Thermal capacitance per unit volume(TIM)	4.0×10^6 $J/(m^3 \cdot K)$

The benchmarks we used are chosen from MediaBench and SPEC2000 benchmark suites, as shown in Table 2. Only the power-intensive applications impose the challenge to the thermal management; so workload type is critical to evaluate the performance of the proposed algorithm. In terms of the power dissipation, the benchmarks are grouped into three categories: hot (power-hungry), cool (non-power-hungry), and warm. The workload patterns used in our experiments is shown in Table 3. The workload is duplicated if needed to continue the simulation. We use M5 simulator [25] to convert each workload pattern/benchmarks into task graphs and extract the execution time and power profile of each job, which is used in equation (10).

Table 2. Benchmarks from Benchmak Suites.

Benchmark Suite	Benchmark
MediaBench	jpegenc, jpegdec mpeg2enc, mpeg2dec
SPEC2000	applu, gcc, bzip2, crc32, mcf, mesa, swim

Table 3. Workload Patterns.

Workload Pattern	Benchmark
HC	jpegdec, gcc
HHM	mpeg2dec, jpegdec, bzip
HHC	jpeg2enc, mpeg2enc, gcc
HMC	jpegenc, applu, gcc
HCC	crc32, mcf, bzip2
HHMM	jpegdec, crc32, mesa, applu
HHCC	jpegenc, jpegdec, bzip2, mcf
HMCHMC	crc32, mesa, gcc, mpeg2enc, applu, mcf
HHMMCC	jpegdec, crc32, mesa swim, mcf, gcc
HHHCCC	jpegdec, jpegenc, crc32, mcf, bzip2, gcc

B.Results and Discussion

In this subsection, we will present our experimental results from five aspects: 1) hot spot; 2) performance; 3) peak temperature; 4) temperature variation; and 5) average temperature. The experimental results with various workload patterns show similar trends in peak temperature, temperature variation and average temperature. Due to space limitation, we only show the results based on workload patterns HMC and HHMMCC.

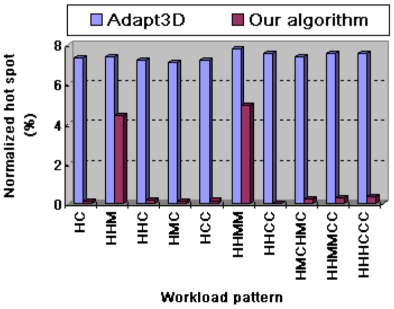


Figure 3. Hot spot comparison.

Hot spot

One of the most important goals for thermal management is to reduce the time of chip operating above the critical temperature. In this part, we evaluate the effectiveness of the proposed algorithm in reducing hot spots. If the on-chip temperature is over the critical temperature, then the chip is regarded as overheated. We report the chip overheated time which is normalized with regards to its total operation time under various workload patterns in Figure 3. When workload contains cool jobs, the proposed algorithm generates negligible hot spots. When workload pattern is either HHM or HHMM, the proposed algorithm generates more hot spots because warm jobs are assigned to the cores with high thermal resistance, which results in hot spots. However, no matter what workload pattern is, the proposed algorithm beats the benchmark algorithm by large margin, as shown in Figure 3.

Performance

In this part, we will evaluate the performance of the proposed algorithm. Performance is measured based on the throughput. Figure 4 presents the normalized throughput of the system based on two algorithms: the proposed algorithm and temperature-aware algorithm. The throughput of the proposed algorithm is normalized with regards to the one of temperature-aware algorithm. Results show that the system boosts throughput up to 7.9% based on our algorithm, compared to Temperature-aware. Heat is easily removed off the chip in our algorithm, and the cores are rarely stalled due to overheating. That is the fundamental reason that our algorithm has higher performance no matter what workload the system executes.

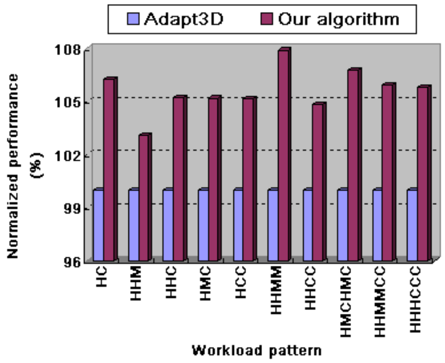
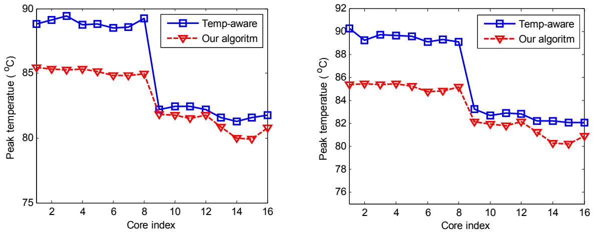


Figure 4. Performance comparison.

Peak temperature

Figure 5 presents the peak temperature of each core under two workload patterns. By observing Figure 5, we find that cores far from the heat sink have high peak temperature in both algorithms; but our algorithm generates lower peak temperature, compared to temperature-aware. The peak temperature of each core in the top silicon layer is about 85^oC based on our algorithm and is about 90^oC based on benchmark algorithm. We observed a peak temperature reduction of 5^oC on average for the cores far from the heat sink and 1.8 ^oC for the cores close to the heat sink, compared to the benchmark algorithm. We also take the average core-level peak temperature over all the 10 workloads. Compared to the temperature-aware algorithm, our algorithm has average peak temperature of 85.4 ^oC, and achieves an average of 4.4 ^oC reduction for cores far from the heat sink, and 1.1 ^oC reduction for cores close to heat sink.



(a) HMC (b) HHMMCC

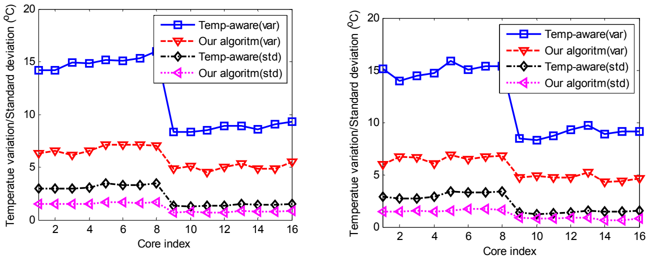
Figure 5. Core-level peak temperature comparison.

Temporal temperature variation

Temporal temperature variation leads to the transient system performance deduction and reliability degrading. To reduce the temperature variation is one of the key issues in the chip design. In this part, we show how the temperature of each core varies based on different scheduling policies. As shown in Figure 6, based on our algorithm the temperature variation magnitude is about 6^oC for the core 1 to core 8 and 4.5 ^oC for core 9 to 16. Based on the benchmark algorithm, the variation is much larger. The variation magnitude is around 15^oC for the cores 1 to 8; and 9^oC for core 9 to 16. Temperature variation that our algorithm generates is about 40% as much as that Temperature-aware does for the core far from the heat sink; about 50% for the cores close to the heat sink. The similar observation exists for the standard deviation of the temperature variation, as shown in Figure 6. In another word, our algorithm generates less temperature variation.

In our algorithm, workload assignment always follows the same pattern: hot jobs go to the cores closer to the heat sink; cool jobs go to the cores farther from the heat sink. Hence the temperature variation is not drastic. In temperature-aware algorithm, hot jobs can go to any core as long as its temperature is low. In the case the hot job is assigned to the core farther from the heat sink, then the temperature of the core goes up quickly, which leads to the large variation, as shown in Figure 6.

Also note that in both algorithms the cores in the top silicon layer have larger temperature variation than the cores in the bottom layer. That is because those cores in the top silicon layer are subject to larger thermal resistance to remove heat off chip.

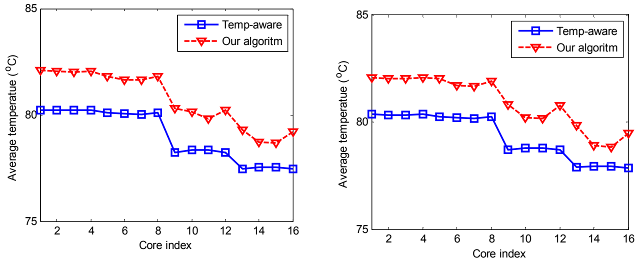


(a) HMC (b) HHMMCC

Figure 6. Core-level temperature variation and standard deviation.

Average temperature

In this part, we present the average temperature of each core. The experimental results show that in our algorithm the core usually has higher average temperature as shown in Figure 7. That is because the system executes more jobs at the same time interval based on our algorithm, as demonstrated by Figure 4. The execution of more jobs produces more heat, which pushes up the average temperature of each core. However, the proposed algorithm is able to control the on-chip temperature below the critical temperature. And the higher average temperature does not generate more hot spots, as shown in Figure 3.



(a) HMC (b) HHMMCC

Figure 7. Core-level average temperature.

VI. Conclusions

In this paper, we have proposed a thermal aware job assignment algorithm for 3D CMP system. In order to transfer heat off the chip more efficiently, the proposed algorithm always assigns hot jobs to the cores closer to the heat sink, and cool jobs to the cores farther from the heat sink. By taking the proposed job assignment strategy, heat stays less time on chip and the chip is able to keep cooler. The experimental results show that the proposed algorithm beats the benchmark algorithm by large margin from four aspects: hot pot, performance, peak temperature and temporal temperature variation. The average temperature of each core in our algorithm is higher because each core executes more jobs at the given time interval, which brings in higher throughput and performance.

References

[1] S. Y. Borkar, P. Dubey, K. C. Kahn, D. J. Kuck, H., “Platform 2015: Intel processor and platform evolution for the next decade”, *Intel Corporation, Tech. Rep*., Mar. 2005

[2] P. Kapur, G. Chandra, K. C. Saraswat, “Power estimation in global interconnects and its reduction using a novel repeater optimization methodology”, *Design Automation Conference*, 2002

[3] W.-L. Hung, G. M. Link, Y.Xie, N. Vijaykrishnan, M. J. Irwin, “Interconnect and thermal-aware floorplanning for 3D microprocessors”, *International Symposium on Quality of Electronic Design*, Mar. 2006.

[4] W. Topol, D.C. La Tulipe Jr., L. Shi, D. J. Frank, K. Bernstein, S. E. Steen, A. Kumar, G. U. Singco, A. M. Young, K.W. Guarini, M. Leong, “Three-dimensional integrated circuits”, *IBM J. Research and Development*, vol. 4, 2006.

[5] B. Black, M. Annavaram, N. Brekelbaum, J. DeVale, L. Jiang, G. H. Loh, D. McCaule, P. Morrow, D. W. Nelson, D. Pantuso, P. Reed, J. Rupley, S. Shankar, J. Shen, C. Webb, “Die stacking (3d) microarchitecture”, *International. Symposium on Microarchitecture*, Dec. 2006, pp.

[6] S. Borkar, “Design Challenges of Technology Scaling”, *IEEE Micro*, 19(4), 1999.

[7] S. Gunther, F. Binns, D. M. Carmean, J. C. Hall, “Managing the Impact of Increasing Microprocessor Power Consumption”, *Intel Technology Journal*, 2001

[8] Failure mechanisms and models for semiconductor devices, JEDEC publication JEP122C. http://www.jedec.org

[9] S. Liu, et al, “Full-chip leakage current estimation based on statistical sampling techniques,“ in *Proc of GLSVLSI*, 2008

[10] S. Liu, et al, “A probabilistic technique for full-chip leakage estimation, “ in Proc of *ISLPED*, 2008

[11] R. Rao, A. Srivastava, D. Blaauw, D. Sylvester “Statistical estimation of leakage current considering inter- and intra-die process variation”, *International Symposium on Low Power Electronics and Design*, 2003.

[12] C. J. Lasance, “Thermally driven reliability issues in microelectronic systems: status-quo and challenges”, *Microelectronics Reliability*, 2003.

[13] D. Brooks, M. Martonosi, “Dynamic Thermal Management for High-Performance Microprocessors”, *International Symposium on High-Performance Computer Architecture*, 2001.

[14] A. Kumar, L. Shang, L. S. Peh, N. K. Jha, “HybDTM: A Coordinated Hardware-Software Approach for Dynamic Thermal Management,” *Design Automation Conference*, 2006.

[15] J. Donald, M. Martonosi, “Techniques for multicore thermal management: Classification and new exploration,” *International Symposium on Computer Architecture*, 2006.

[16] J. Cong, J. Wei, and Y. Zhang, “A thermal-driven floorplanning algorithm for 3D ICs”, *International Conference on. Computer-Aided Design*, 2004,

[17] B. Goplen, S. Sapatnekar, “Efficient thermal placement of standard cells in 3D ICs using a force directed approach”, *International Conference on Computer-Aided Design*, 2003.

[18] K. Puttaswamy, G. H. Low., “Thermal Herding: Microarchitecture Techniques for Controlling HotSpots in High-Performance 3D-Integrated Processors”, *IEEE International Symposium on High Performance Computer Architecture*, Feb. 2007.

[19] C. Zhu, C. Zhu; Z. Gu, L. Shang, R.P. Dick, R. Joseph, “Three-dimensional chip-multiprocessor run-time thermal management”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2008.

[20] C. Sun, L. Shang, R. P. Dick, “3d multiprocessor system-on-chip thermal optimization”, *International Conference on Hardware Software Codesign*, 2007.

[21] A. K. Coskun, J. L. Ayala, D. Atienza, T. Simunic, Y. Leblebici, “Dynamic Thermal Management in 3D Multicore Architectures”, *Design, Automation & Test in Europe*, Apr. 2009.

[22] X. Zhou, Y. Xu, Y. Du, Y. Zhang, J. Yang , “ Thermal Management for 3D Processors via Task Scheduling”, *International Conference on Parallel Processing*, Sep. 2008:

[23] K. Skadron, M. R. Stan, K. Sankaranarayanan, W. Huang, S. Velusamy, D. Tarjan, “Temperature-aware microarchitecture: Modeling and implementation”, *ACM Transactions on Architecture and Code Optimization***,** 2004

[24] K. A. Bowman, B. L. Austin, J.C. Eble, X. Tang, J. D. Meindl, “A physical alpha-power law MOSFET model”, *IEEE J. Solid-State Circuits,* vol. 34, Oct. 1999.

[25] N. L. Binkert, R.G. Dreslinski, L.R. Hsu, K. T. Lim, A. G.. Saidi, and S. K. Reinhardt, “The M5 simulator: Modeling networked systems”, *Proc. IEEE Micro Special issue on Architecture Simulation and Modeling*, Jul. 2006