Area-Efficient Microarchitecture for Reinforcement of Turbo Mode

Shinobu MIWA†, Member, Takara INOUE†, Nonmember, and Hiroshi NAKAMURA†, Member

SUMMARY Turbo mode, which accelerates many applications without major change of existing systems, is widely used in commercial processors. Since time duration or powerfulness of turbo mode depends on peak temperature of a processor chip, reducing the peak temperature can reinforce turbo mode. This paper presents that adding small amount of hardware allows microprocessors to reduce the peak temperature drastically and then to reinforce turbo mode successfully. Our approach is to find out a few small units that become heat sources in a processor and to appropriately duplicate them for reduction of their power density. By duplicating the limited units and using the copies evenly, the processor can show significant performance improvement while achieving area-efficiency. The experimental result shows that the proposed method achieves up to 14.5% of performance improvement in exchange for 2.8% of area increase.

key words: microprocessors, turbo mode, chip temperature, design methodology

1. Introduction

Chip temperature is one of the key factors to characterize processor performance in modern computer systems. To protect themselves from serious damage such as solder peel-off, commercial processors are designed to maintain their peak temperature below a certain limit. Since turbo mode, which accelerates many applications without major change of existing systems, allows processors to consume much more power, processors during turbo mode often hit the above temperature limit if they use aggressive frequency and voltage. When the chip temperature reaches its limit, a processor stops the acceleration and dynamic thermal management (DTM) such as clock gating and DVFS then starts to work for cooling [1]. The available frequency during turbo mode has to be lowered for the avoidance of the thermal violation and it constrains the capability of turbo mode.

This paper presents that adding small amount of hardware enables microprocessors to significantly reinforce turbo mode. As above mentioned, turbo mode stops when the peak temperature reaches a certain threshold. In other words, if the peak temperature never exceeds the threshold, turbo mode can continue permanently. Heat depends on power consumption, and dynamic power consumption depends on switching activity of transistors. Therefore, duplicating units that become heat sources and shifting activity from one to another enable a processor chip to keep the peak temperature a lower level. As a result, the processor chip can maintain turbo mode for a longer time or utilize more aggressive clock frequency during turbo mode.

The proposed method may be regarded as a kind of DTM, but it is different from conventional DTM in terms of both its purpose and requirement. The main concern of the conventional DTM is to protect processors against critical damage [2]–[4]. Conventional thermal management methods therefore start to work just when the peak temperature of a chip reaches a dangerous level, and stop when the chip becomes cool. That is, conventional DTM operations are accidental and temporal. In this case, immediate reduction of chip temperature is more important than preservation of processor performance. On the other hand, our purpose is to extend acceleration time or to realize more aggressive acceleration (i.e. to use higher frequency and voltage during turbo mode). In our case, thermal management methods always work during the acceleration (it means that our thermal management works preventively [5]), so the requirement of the methods is to have few performance penalties.

Actually, this paper is not the first one to present such a preventative thermal management method and its usage. The initial concept of our method was presented by Heo et al. in 2003 [6], which is called activity migration (AM). Thereafter, a type of AM has been studied by Chaparro et al. [7], [8], but neither Heo nor Chaparro pay attention to area optimization of processors with AM despite its importance. Although they apply AM to several coarse-grained modules (e.g. a bundle of register files and execution units, a cluster [9], and all units) within a processor core, fine-grained AM such as AM for individual ALU has not been studied yet. Moreover, previous work empirically determines target modules of AM and their duplication count. How to determine migration targets and their duplication count for the future microprocessors is still an open question for computer architects.

This paper proposes spatially fine-grained AM and the design methodology of microprocessors using it. As described later, limited modules within a processor heat up the processor chip, so duplicating only these modules could achieve higher area-efficiency while preserving performance improvement caused by AM. Our design method is to detect such modules in architecture design stages and then to duplicate them by the appropriate number. For this purpose, we develop a new thermal model of modules.

The major contributions of this paper are summarized as follows.

---

†The authors are with The University of Tokyo, Tokyo, 113–8656 Japan.
†∗Presently, with Ryoyu Systems Co. Ltd.
a) E-mail: miwa@hal.ipc.i.u-tokyo.ac.jp
DOI: 10.1587/transinf.E97.D.1196

Copyright © 2014 The Institute of Electronics, Information and Communication Engineers
This is the first paper of spatially fine-grained AM. As described later, some limited modules become heat sources and their area overhead is quite small. This fact indicates that fine-grained AM is a promising approach to achieve both performance improvement and area-efficiency.

We present how to determine candidates of AM and their duplication count in an architectural design stage. In contrast to empirical approaches, our method can appropriately select migration targets and their duplication count for the future processors whose temperature is unknown.

To our best knowledge, this paper first evaluates area and performance of processors with coarse-grained AM thoroughly. That is, fair comparison between them is first provided. Comparing to these processors, our processors achieve higher performance while lowering area increase.

The rest of this paper is organized as follows. First of all, we introduce turbo mode in the view of chip temperature. Next, Sect. 3 overviews AM and summarizes the previous approaches. Section 4 refers to motivation of our work. More specifically, this section mentions importance of heat source detection in fine-grained AM and its difficulty. Next, Sect. 5 provides the detailed explanation about the proposed method to design area-efficient processors with fine-grained AM while mainly focusing on our thermal model. The experimental setup is summarized in Sect. 6, and then the experimental results are presented in Sect. 7. We overview related work in Sect. 8, and finally Sect. 9 concludes this work.

2. Turbo Mode and Chip Temperature

Modern chip multiprocessors (CMPs) provide turbo mode that has capability to accelerate many applications. When some cores are inactive in a CMP, the processor can change its state to turbo mode that allows to increase clock speed of the rest. Turbo mode enables microprocessors to accelerate many applications without major change of existing hardware and compilers, so it is widely used in commercial processors [1], [10].

According to the Intel’s white paper [1], turbo mode is constrained by the following factors:

- power delivery limits of systems
- the estimated current consumption of the processor
- the estimated power consumption of the processor
- the temperature of the processor

The first three factors are not serious problems for use of aggressive clock frequency and voltage because current computer systems still have sufficient headrooms of them. This is substantiated by some results of over-clocking. On the other hand, the temperature limit is crucial because the cooling capability of current computer systems can not tolerate use of excessively high frequency and voltage. Special cooling mechanisms like liquid helium are needed for further over-clocking. Therefore, this paper focuses on the temperature limit of turbo mode.

Processors during turbo mode satisfy the temperature limit in cooperation with DTM. A processor in an active state (i.e. P0 state in ACPI[11]) steps up its clock frequency and voltage to the predetermined level if turbo mode is enabled. The available frequency during turbo mode is factory-configured and dependent on the number of active cores. Over-clocking continues while the chip temperature is under its constraint. When the peak temperature is close to a dangerous level, the processor stops the acceleration and DTM then starts to work for cooling. After the chip temperature is lowered to a certain level, the processor restarts to boost the clock speed [1].

To demonstrate how the chip temperature affects the acceleration capability of turbo mode, we evaluated performance degradation caused by stopping a CPU fan during turbo mode. Table 1 summarizes the configuration of the experimental machine. The CPU is Core i7-3930K that has 6 cores. The CPU frequency is typically 3.2 GHz, while it steps up to 3.8 GHz during turbo mode. ZALMAN CNPS9900 MAX shown in Fig. 1 was used as a CPU cooler. We evaluated the situation that running a single-thread program was boosted by turbo mode. Twenty programs in SPEC CPU 2006 benchmark suits were used for the evaluation. All programs were compiled with gcc-4.3.3 with the “O3” option. The reason for use of the other programs in the benchmark suits is that they are too short to measure CPU frequency or cannot be compiled in our system. CPU frequency during turbo mode was measured with i7z that reports average CPU frequency of each core per one second [12]. The OS is Ubuntu-10.04.

Figure 2 shows CPU frequency during turbo mode. The horizontal axis represents programs, while the vertical

| Parameters | Remarks |
|------------|---------|
| CPU        | Core i7-3930K (6-core, 130 W TDP) |
| CPU Frequency | 3.2 GHz (normal)/3.8 GHz (maximum) |
| Memory     | 16 GB DDR3 SDRAM |
| CPU cooler | ZALMAN CNPS9900 MAX |

Fig. 1 A cooling device to make the most out of turbo mode's ability in Intel Core i7-3930K.
axis represents average CPU frequency during turbo mode. The left bars indicate the case of using the CPU fan, while the right bars indicate the case of stopping the fan.

As shown in the figure, the processor spends the most of time working at 3.7 GHz when the CPU fan is used. That is, the processor hardly faces the temperature limit even during turbo mode. In contrast, the processor without the fan often hits the temperature limit and therefore must lower its frequency. As a result, it is difficult for the processor to maintain the normal CPU frequency (3.2 GHz) for the most of workloads. CPU-bound applications such as 416.gamess and 464.h264ref frequently violates the limit, while memory-bound applications such as 429.mcf and 433.milc infrequently does.

Figure 3 presents the comparison result of elapsed time of two processors. As shown in the figure, stopping the CPU fan worsens elapsed time by up to 32.6% depending on applications. Therefore, it is important for processor designers to balance between the available frequency of a processor and the capability of its cooling system.

As previously mentioned, lowering the chip temperature during turbo mode has two merits. One is to allow processors to use more higher frequency. Since processors whose temperature is low hardly hit the temperature limit during turbo mode, they can boost their CPU clock more aggressively. The CPU frequency greater than 3.8 GHz could be acceptable for “/w fan” if its chip temperature was reduced. The most of CPUs would benefit from this re-tune of available CPU frequency. The other is to extend duration of turbo mode. Lowering the chip temperature reduces the thermal violation under the certain frequency. It would improve the performance of CPUs under the stringent thermal constraint like “/wo fan.” However, this is a rare situation, so this paper focuses on the former one.

3. Activity Migration

This section provides the detailed explanation of AM. First, we summarize its mechanism, the merits and demerits. Next, we refer to what units are selected as migration candidates and how to select them in previous work. Finally, we summarize impact of spatial granularity of AM.

3.1 Overview

Figure 4 illustrates block diagrams of a processor using AM. In the figure, execution units and a register file are regarded as a migration target. As shown in the figure, the processor with AM is equipped with N copies of the target unit. In contrast to the case of clustered processors [9], copies and the original are never used concurrently. The figure expresses inactive units as shaded ones.

The processor with AM executes a program normally unless the temperature of the target unit reaches a predetermined value. Figure 4 (a) represents this situation. In the figure, the unit $M_0$ is used for computation, while the others are not. This situation is equivalent to the situation where a processor without AM executes this program. The temperature of the active unit gradually rises through execution.

When the temperature reaches the threshold, the processor stops its pipeline and switches from the above unit to another. Instead of temperature-based switching, periodical switching is often used [6]. In Fig. 4 (b), the unit $M_1$ is selected as the next one. Then, some processor states such as register values are copied from the original to the next.
After the above process is complete, the processor resumes the execution using the next unit. The original gradually cools because the processor stops using it. In this manner, processors with AM can reduce the peak temperature of units while maintaining computation.

Figure 5 demonstrates temperature trend of the units. As shown in Fig. 5 (a), the temperature gradually rise while a processor is using these units continuously. The temperature rise eventually stops at some temperature if it is under a threshold of DTM ($T_{\text{limit}}$). On the other hand, a processor with AM switches between the duplicated units before their temperature reaches the saturated temperature (Fig. 5 (b)). As a result, AM can reduce the peak temperature to a certain level ($T_{\text{high}}$).

Since AM generates a larger margin between the peak temperature and its limit, processors with AM can utilize higher clock frequency and supply voltage. The original paper shows that AM can increase clock frequency of a ring oscillator by up to 15.9%, but how much processors with AM can upgrade the clock frequency is not evaluated [6].

AM is a technique to reduce dynamic power consumption of units, but it is easily extended to reduce power consumption including static one. For reduction of static power consumption, unused units like $M_1$ in Fig. 4 (a) should be powered off. However, it requires the modification of power domain and results in area increase. Clarifying the impact of static power reduction by AM is interesting, but it beyonds the scope of this paper. Therefore, this paper regards AM as the original one.

Although AM is a good solution to keeping chip temperature low, it has two major drawbacks listed below.

**Performance penalty** AM degrades IPC for twofold reasons: migration overhead and long interconnects.

To migrate activity between copies, the processor must drain instructions from the current unit and copy some processor states to the next one. This process takes dozens of cycles depending on units. Therefore, frequently switching damages processor performance.

Adding copies may cause dramatic change of chip layout and then lengthen the distance between cooperating units. That is, longer wires are needed for their connection. To maintain clock frequency, further pipelining may be required against long wires, but it hurts IPC.

**Area overhead** Obviously, AM enlarges chip area. Although future technology scaling is expected to yield a lot of transistors we cannot use [13]–[18], saving area is still important in the future processors. If a processor with AM is realized compactly, the remaining area can be used for another purpose such as implementation of specialized hardware [14], [16]–[18].

### 3.2 Previous Design of Processors with AM and Its Issues

In previous work, targets of AM and their copy count is determined empirically. That is, units known as hot spots are selected as migration candidates and duplicated by a few copies. Moreover, the migration candidates are coarse-grained like fetcher or renamer. Although fine-grained AM could realize area-efficient high-performance processors, nobody evaluates performance and area of processors with fine-grained AM.

Heo et al. propose four types of AM: inter-core; inter-core excluding an L1D cache and TLB; inter-core excluding the above units plus a fetcher; and interset of register files and execution units [6]. All migration targets are doubled in their approach. They compared IPC degradation of four types, and intended that the interset migration was best in terms of area-efficiency. However, they did not evaluate temperature reduction and clock frequency improvement on these processors.

Chaparro et al. applied AM to clustered superscalar processors [7], [8]. They show that inter-cluster AM between two clusters can reduce the peak chip temperature by up to 33% in exchange for 14% of IPC degradation. However, these papers do not present how much performance improvement can be achieved by upgrading clock frequency. In addition, the papers do not compare their approach with Heo’s approach, so it is unknown that inter-cluster AM can really achieve both high performance and area-efficiency.

It is hard for empirical approaches to realize area-efficient processors with fine-grained AM because units that do not become heat sources may be selected as migration candidates. In addition, deciding duplication count for each migration candidate is a difficult problem in these approach. The brute force search is needed for finding optimal migration candidates and duplication count, but it becomes tremendous work as migration targets are getting fine. We will provide the detailed explanation about it in Sect. 4.

### 3.3 Spatial Granularity of AM

Comparing to spatially coarse-grained AM, spatially fine-grained AM has less demerits. By selecting migration targets appropriately, fine-grained AM can realize a processor whose peak temperature is more flattened than coarse-
grained AM. Moreover, such processor shows less area increase because fine-grained AM can avoid duplication unnecessary for peak temperature reduction.

The unique disadvantage of fine-grained AM is that it may lengthen more critical wires connecting cooperating units. For example, if an ALU is selected as a migration target, total area of ALUs is enlarged by its duplication. It means that the distance between ALUs is further and then bypassing operands between them within one cycle may become difficult. Moreover, the register access latency may increase because some ALUs locate in the place far from register files.

Generally speaking, more critical wires are extended as migration granularity is finer. As a result, a lot of critical paths need pipelining. For example, although we can consider AM within an ALU as a further fine-grained AM, it would incur pipelining of ALUs and then hurt IPC largely.

Actually, the issue of operand bypasses caused by duplication of ALUs can be alleviated by appropriate selection of concurrently used set of ALUs. Physically adjacent ALUs should be used as a pair. Since bypasses between ALUs unused concurrently are not needed, long bypasses can be eliminated in this manner. Moreover, pipelining critical units is not needed in unit-level AM as shown in Sect. 6. Therefore, this paper focuses on unit-level AM.

4. Motivation

Hot units do not match heat sources entirely. For example, it is well-known that ALUs and register files are hot spots in microprocessors [6], [19], but these units do not generate heat uniformly. Some units yield large amount of heat, while the others do not. Therefore, for area-efficiency, only the former units should be regarded as candidates of AM.

Figure 6 presents peak temperature and cumulative area of units within a processor. The detailed simulation methodologies including the processor configuration are described in Sect. 6. The horizontal axis represents unit names, while the vertical axis represents peak temperature or cumulative area. The bars indicate peak temperature, while the dotted line indicates cumulative area.

The graph presents that some units show similar high temperature. The peak temperature of seven units (from alu.1 to BTB) is within only 1 degree Celsius, while that of eleven units (from alu.1 to ROB) is within 2 degrees Celsius. If all these units are regarded as migration candidates, the area overhead of AM is not negligible because area of the former unit group and the latter one is 10.8% and 17.1% of core area, respectively. That is, if all these hardware is doubled, the area overhead reaches 10.8% and 17.1% in each.

Note that all hot units do not generate large amount of heat. Figure 7 presents thermal map of the above processor when it finishes the execution of 456.hmmer in SPEC CPU 2006 benchmark program suits. The thermal map implies that although intREG (Integer Register File) and BTB show high temperature, these high temperature may be incurred by heat transfer from surrounding units. BTB looks like a unit heated up by ifRAT (Integer Front-end Register Alias Table), while intREG looks like a unit heated up by ifRAT and alu.0.

As previously mentioned, AM can reduce power consumption of target units by reducing their use, and then their peak temperature is lowered. Therefore, duplicating units heated up by surrounding units make less contribution to reduction of the peak chip temperature. It wastes chip area. Thus, to achieve both high performance and area-efficiency on processors with fine-grained AM, it is important to detect units that become heat sources such as ifRAT and to select only them as migration candidates.

Empirical approaches, which regards well-known hot units as migration candidates, cannot correctly detect the units that become heat sources. They may select vain units as migration candidates. Moreover, if we do not know which units become hot, we cannot select migration targets in these approaches. This situation often happens in processor development. In the architecture design stage, we often try to implement a quite new unit for a newly developed function on a processor, but we do not know its peak temperature. In addition, it is hard for these approaches to find the optimal
duplication count for each migration candidate because we do not know how much temperature reduction a duplication count causes. The brute force search may be able to find the optimal migration candidates and their optimal duplication count in coarse-grained AM, but it becomes harder as the migration granularity is getting fine because of its broad search space. Therefore, the methods to detect heat source units and to determine their appropriate duplication count are needed for area-efficient AM.

5. Design of Area-Efficient Microarchitecture with Fine-Grained Activity Migration

This section describes a design methodology for area-efficient microprocessors with AM. First, its design flow is presented. We determine migration targets and their duplication count according to the thermal model we develop. So, we secondly provide the detailed explanation about the model, and thirdly mention how to decide the duplication count. Finally, we discuss the proposed method for deep understanding.

5.1 Design Flow

The proposed design flow to realize area-efficient processors with AM consists of the following three steps.

1. design architecture of a base processor
2. detect heat sources in the above processor and decide their copy count
3. reflect the above copies in the design

First step is to design architecture of a base processor which does not utilize AM. This process is as usual as conventional architecture design. We design the desired architecture with some processor simulators such as performance [20]–[22], power, area [23], and sometimes thermal ones [24].

After the baseline architecture design finishes, we find major heat sources from the base processor and decide their copy count. We use a new thermal model for this process. The model is detailed in Sect. 5.2. The algorithm of heat source detection and copy count decision is described in Sect. 5.3. We can know the appropriate copies in this manner.

Finally, we reflect the above copies in the current architecture design. Then, we can continue usual design flow of processors (i.e. we start the circuit design).

5.2 Thermal Model to Design Area-Efficient Microprocessors with Fine-Grained AM

For heat source detection, we have to know how potential energy for raising temperature each unit has. To this end, we estimate the temperature rise of a unit if only the unit exists in the chip, as shown in Fig. 8. By excluding the impact of heat transfer from surrounding units, we can know how much the unit has capability of heating the chip up.

Temperature $T$ of a material in a steady state is defined as follows.

$$T = R \times P + T_{amb}$$

where $R$ is thermal resistance of the material, $P$ is peak power consumption of the material, and $T_{amb}$ is ambient temperature. That is, the above model provides conservative estimation about the temperature of the material.

Here, a target unit and surrounding ones can be regarded as independent materials. Under this assumption, $T_{amb}$ means temperature of surrounding units, while $R \times P$ means temperature rise caused by heat the unit itself generates. Thus, we can regard that the units of which $R \times P$ is large are heat sources.

Thermal resistance $R_u$ for a unit can be described as follows [19], [24].

$$R_u = \frac{1}{1/(R_{conv} + R_{hs} + R_{sp} + R_{v}) + 4/(R_{lu})}$$

where $R_{conv}$ is convective thermal resistance; $R_{hs}$ is thermal resistance of heat sink; $R_{sp}$ is thermal resistance of heat spreader; $R_{v}$ is vertical thermal resistance of the unit; and $R_{lu}$ is lateral thermal resistance of the unit (see Fig. 8).

$R_{conv}$ can be expressed by the following equation [25].

$$R_{conv} = \frac{1}{h \times A_u}$$

where $h$ is constant heat transfer coefficient (≠ 0.0012) and $A_u$ is area of the unit.

$R_{hs}$, $R_{sp}$ and $R_{v}$ are vertical thermal resistance of each material (i.e. heat sink, heat spreader and silicon). Vertical thermal resistance $R_{ver}$ for a material can be described below.

$$R_{ver} = \frac{t}{k \times A_u}$$

where $t$ is thickness of the material and $k$ is thermal conductivity of the material per unit volume.

In contrast, modeling lateral thermal resistance $R_{lu}$ for the unit is quite difficult because processor chips are very thin under the modern LSI technology. Here, we use the simple assumption that each unit is a square and that $R_{lu}$ is inversely proportional to cross-sectional area of the unit. That is, by using a positive constant value $c$, $R_{lu}$ can be described as follows.

$$R_{lu} = \frac{c}{t \times sqrt(A_u)}$$
This equation means that a larger unit has capability to conduct larger amount of heat to surrounding units.

From the above equations, temperature rise $T_{self} (\approx R \times P)$ caused by self-heat can be described below.

$$T_{self} = \left( \alpha + \frac{1}{\beta \times \sqrt{r(A_u)}} + \gamma \right) \times \frac{P_n}{A_u} \tag{6}$$

where $P_n$ is peak power consumption of the unit; and $\alpha$, $\beta$ and $\gamma$ are positive constant values. These constant values can be obtained by curve fitting.

Note that migrating activity contributes only to reduction of dynamic power of a unit. Static power of the unit cannot be saved only if a processor stops using the unit. Therefore, when determining duplication count, we focus on temperature rise caused by dynamic power consumption of the unit. From Eq. (6), using $P_u^{dyn}$ that is peak dynamic power consumption of the unit, we can define $T_{self}^{dyn}$ (contribution of dynamic power to $T_{self}$) as follows.

$$T_{self}^{dyn} = \left( \alpha + \frac{1}{\beta \times \sqrt{r(A_u)}} + \gamma \right) \times \frac{P_u^{dyn}}{A_u} \tag{7}$$

Similar to $T_{self}^{dyn}$, we can define temperature rise caused by static power consumption of the unit, which is called $T_{self}^{st}$. In summary, temperature rise caused by unit’s own heat is proportional to power density of the unit, and inversely proportional to length of a side of the unit. Power and area of each unit for a designing processor can be estimated by a power and area simulator [23]. Thus, using this equation in an architecture design stage, we can estimate the potential capability to rise temperature for each unit.

5.3 Heat Source Detection and Copy Count Decision

We search heat sources and their appropriate copy count in a step-by-step manner. The procedure of detecting heat sources and deciding their appropriate copy count is shown in Fig. 9. The figure illustrates the situation that copy count decision is carried out for a processor that consists of three units (Unit A, B and C). The black bars represent $T_{self}^{dyn}$ of units, while the white bars represents $T_{self}^{st}$.

First, we estimate $T_{self}^{st}$ for all units using the aforementioned model and sort the units according to the estimated values (Fig. 9(a)). As a result, we can know the most dominant heat source in this configuration. In Fig. 9(a), Unit A is such a unit because its $T_{self}^{st}$ is highest. Then, the unit can be regarded as a migration candidate and we make its single copy.

Next, the above procedure is carried out for the new configuration. Since original units and their copies are used evenly under ideal AM, $T_{self}^{st}$ of a unit becomes one-$N$th if $N-1$ copies are installed. Note that $T_{self}^{st}$ is maintained after duplication because original AM has no capability to reduce the static power consumption. As a result, Unit B becomes the most dominant heat source on the processor that doubles Unit A (Fig. 9(b)). Therefore, Unit B is the next migration candidate and is doubled.

Note that the number of copies increases one by one in our method. For example, both Unit A and its copy are the most dominant heat source in the situation (c), but they are originally one. Therefore, the dynamic power consumption of the copy is also reduced if another copy of Unit A is installed (Fig. 9(d)). To avoid over-installation, copies are sequentially added to the base processor.

After the above procedure is repeated several times, we can obtain various configurations that contains different types of copies and their count in the base processor. Then, we carry out the detailed simulation for all configurations to select the best one in terms of high performance and area-efficiency.

Although the proposed method requires the detailed simulation in the final step, it would not be costly in practice. The number of configurations we should test is small because a few units are dominant heat sources in a modern processor as shown in Sect. 6. In addition, adding copies abruptly worsens area-efficiency of processors. As shown in Sect. 7, repeating once or twice is sufficient to find the area-efficient configuration for a modern processor.

5.4 Discussion

The proposed method is a technique to find a sub-optimal solution about migration targets and their copy count. Our method could find the minimum processor configuration if heat transfer from surrounding units was ignorable and our model had significant accuracy. The above assumption is not true in practice, but the ultimate in fine-grained AM is that all units generate heat uniformly. The proposed method tunes processors to go toward this ideal state.

Note that temperature reduction does not directly improve processor performance because of IPC degradation mentioned in Sect. 3. However, reducing the peak temperature incurs significant performance improvement when area increase caused by AM is small, because adding a few small copies to a processor hardly worsens its pipeline depth. In addition, as described later, the impact of migration overhead on IPC is trivial in practice. Therefore, our heuristic that focuses only on the peak temperature reduction would be effective if area increase is small.

In our design methodology, we can use some thermal simulator instead of our thermal model. However, if a thermal simulator is used for the estimation of temperature rise, we must carry out the simulation numerous times. Sup-
pose that a core has 30 units and eight processes runs on eight boosted cores. To estimate the temperature rise of each unit within boosted cores, 240 times thermal simulation is needed. Simulation count increases as boosted cores increase, and it makes the above simulation a troublesome job. For the fast search of migration targets and copy count, we use the thermal model.

6. Experimental Setup

First, three constant values of the proposed model was computed by curve fitting. We placed a square unit at the center of a chip, and collected sample data as its area and power density changed. The area changed from 0.403 to 3.14 mm², while the power density changed from 0.2 to 1.4 W/mm². For each combination of area and power density, the temperature in a steady state was measured by HotSpot. The simulation parameters including chip conditions are listed in Table 2. This procedure resulted in 58 sample data, and then we trained our model with them. As a result, we obtained the following parameters: \( \alpha = 86.24, \beta = 0.00075 \) and \( \gamma = 0.011364 \).

The result of curve fitting is shown in Fig. 10. The horizontal axis represents area, while the vertical axis represents temperature. The legend represents power density of each unit. The gray lines indicate \( T_{\text{self}} \) calculated by the proposed model, while the dots indicate the sample data obtained from HotSpot. The graph presents that our equation can nicely model the temperature rise of a single heat source in a chip. The MSE of the proposed model is 0.618.

Next, using this model, we designed processors with fine-grained AM against a base processor (BASE). The parameters of the base processor are summarized in Table 3. The base processor is single-core because this paper focuses on the situation that a processor enters turbo mode during single-thread program execution. The CPU frequency in the normal state is 3 GHz.

Power consumption and area of each unit within the base processor were calculated by the modified version of McPAT 0.8 [23] and Onikiri2 [22]. McPAT is a processor simulator that can compute area and peak power consumption of units, but the original codes contain several bugs (e.g., evaluation function of memory arrays and estimation of load capacity of decoders) so we corrected them for this experiment. In addition, McPAT overestimates the peak power consumption of each unit because it assumes that all transistors within the unit are switched. To alleviate this overestimation, activity factor of each unit is needed. For this purpose, we used a cycle-accurate processor simulator, Onikiri2. We modified Onikiri2 to calculate the activity factors every 1M cycles, and then carried out the simulation with SPEC CPU 2006 benchmark suits [26]. In this manner, we can know the highest activity factor for each unit. The peak power consumption of a unit was computed by the multiplication of the above value by the peak power consumption of the unit McPAT calculated.

Figure 11 shows \( T_{\text{self}} \) of each unit in the base processor. The horizontal axis represents unit names, while the vertical axis represents temperature. The black bars represent \( T_{\text{self}}^{\text{dyn}} \), while the white bars represent \( T_{\text{self}}^{\text{st}} \). Since IPC for some programs exceeds 2, alu_0 and alu_1 are sometimes used in the same manner. As a result, two ALUs show the similar power density in the worst case, so we regard them as a single migration candidate.

As shown in the figure, top three units show large temperature rise: iRAT, the unified ALUs and fRAT (Floating-point Front-end RAT). Their temperature rise is 10.9, 7.33 and 6.07 degree Celsius, respectively. For this processor, we twice repeated the procedure described in Sect. 5.3 and then obtained two processors for fine-grained AM (PRO1 and PRO2), which are listed in Table 4.
Fig. 11  Estimation result of $T_{sel}$ in the base processor.

**Table 4  Processors with fine-grained AM.**

| Name  | Remarks                                                                 |
|-------|--------------------------------------------------------------------------|
| PRO1  | with one copy of ifRAT (the first configuration in the algorithm of Sect. 5.3) |
| PRO2  | with one copy of ifRAT and the unified ALUs in each                      |
|       | (the second configuration in the algorithm of Sect. 5.3)                  |

**Table 5  Coarse-grained AM.**

| Name  | Remarks                                                                 |
|-------|--------------------------------------------------------------------------|
| AM-D  | Register files (INT and FP), four ALUs,                                 |
|       | multiplier and FPU are doubled                                           |
| CLST  | Schedulers (INT/FP and load/store queues) and D-Cache are doubled in addition to AM-D |

We compared these processors with existing designs with coarse-grained AM. From the literature [6], we used AM that doubles the interset of all execution units and all register files, because the authors intend that this AM is best considering area-efficiency. We name this processor as AM-D. From the literature [8], we used inter-cluster AM. To maintain IPC, we doubled all units in back-end of the base processor instead of clustering them. We name the processor designed in this manner as CLST. The units duplicated in these coarse-grained AM are summarized in Table 5.

For the thermal simulation, we designed the floorplans of the above processors. Since the floorplan of the base processor is already presented in Fig. 7, the other floorplans are shown in Figs. 12 to 15. Note that the duplicated units are colored in the figures. Based on the floorplan of a 4-core Nehalem processor [27], we designed these floorplans except for CLST. The floorplan of CLST were made from one presented in the literature [8] considering to the layout difference of base processors. All units’ area was computed by McPAT.

The impact of extended interconnects was considered as follows. In floorplanning, the interconnect length is usually defined as Manhattan distance between two units. Therefore, we first searched the longest Manhattan distance between cooperating units in the base processor and found that the path from ifRAT to FpQ (Floating-point Instruction Queue) was longest. We regard that half of dispatch latency is delay of this interconnect and that this is the limit length of wires transmitting signals within one cycle. Next, for floorplans shown in Figs. 12 to 15, all extended interconnects (i.e. wires whose Manhattan distance is lengthened) was compared with the above limit. If the length of an interconnect is beyond the limit, we regard that it needs additional pipelining. In this manner, we calculated penalties of all extended interconnects. The result is summarized at the second column in Table 6.
Table 6  Pipeline depth and migration overhead in processors with AM.

| Name | Additional stage | Migration overhead |
|------|------------------|--------------------|
| PRO1 | rename +1        | 9 cycles stall      |
| PRO2 | rename +1, IntExec +1 | 9 cycles stall, reschedule |
| AM-D | dispatch +1, issue +1 | 33 cycles stall, reschedule |
| CLST | dispatch +4, IntExec +1 | 33 cycles stall, reschedule, D-Cache flush |

The last column of the table summarizes migration overhead of each processor. As shown in the table, the processors that require migration of RAT (PRO1 and PRO2) stall 9 cycles every migration, because each RAT consists of a single-cycle 32-entry CAM that can accept 12-read and 4-write operations concurrently. In the processors that need to stop their back-end pipeline (PRO2 and below), all instructions already issued but not completed are flushed when migration is carried out, and they are rescheduled after the migration is complete. It takes 33 cycles for copying register values in the processors that require migration of register files (AM-D and CLST), because our register files are composed of single-cycle 128-entry RAMs that enable 12-read and 4-write operations to execute. In addition to the above penalties, CLST needs D-Cache flush as described in the literature [8].

The CPU frequency available during turbo mode is usually determined through thermal stress tests. Processor designers run CPU stress programs on the designed processor in various situations (e.g. frequency, cooling devices, ambient temperature, etc.) and then select the best setting as the recommended one. As shown in Sect. 2, the chip temperature at the available frequency generally has a significant headroom in a commercial processor. This is because commercial processors are designed not to violate their temperature limit in a typical use case (even during turbo mode). For this purpose, processor designers determine the available frequency to provide some headroom in the chip temperature. This margin was regarded as a tuning parameter in our experiment.

We used the most CPU-bound program (456.hmmer) in our benchmark suit as a CPU stress program, and then designed two settings per a processor, which have different available frequency (i.e. different temperature margins). The temperature limit of processors was regarded as 90 degree Celsius [28], and 10 and 15 degree Celsius were used as the temperature margins. This paper evaluates the performance of five processors under the above settings.

To investigate the maximum clock frequency under the thermal constraint (i.e. 75 and 80 degree Celsius), we carried out the experiment illustrated in Fig. 16. First, we set some clock frequency and supply voltage. Then, we carry out performance simulation with Onikiri2 to obtain IPC and activity factor traces of units for an application. In parallel, power simulation for the processor with the above frequency is carried out using McPAT. From these simulation results, we create power traces of units, which are used for thermal simulation. After the thermal simulation finishes, we can know the peak chip temperature when this application runs on a processor with this set of frequency and voltage. These procedures are repeated unless the peak temperature exceeds the thermal constraint. When the peak temperature is beyond the thermal constraint, the clock frequency used in the previous iteration can be regarded as the maximum under the thermal constraint.

The other simulation conditions are summarized as follows. We use a linear function of clock frequency as scaling law of supply voltage. More specifically, the voltage is calculated by $V = 0.19F + 0.62$ (note that $V$ and $F$ are voltage and frequency, respectively). We confirmed that this equation had significant accuracy by using a real Sandy-Bridge machine. We used 24 out of 29 programs in SPEC
CPU 2006 benchmark suits for the experiment, because 5 programs were aborted during the performance simulation. First 1G cycles were skipped, and the following 6G cycles were used for the evaluation. The reason for use of SimPoint [29] is that the thermal simulation requires continuous power traces with certain length. We increased the memory access cycles shown in Table 3 depending on the CPU frequency, because turbo mode only changes the CPU frequency and voltage. In our experiment, migration intervals that are equivalent to sampling intervals of traces were changed from 100K to 10M cycles, which are also used in the literature [6]. Thus, the stall rate of processors with AM is a negligible level that is up to 0.033%.

7. Experimental Result

Table 7 shows the sets of available CPU frequency for five processors under two temperature margins. The second column represents the available frequency under 10 degree Celsius of the margin, while the third column represents the available frequency under 15 degree Celsius of the margin. The available frequency shown in the table is the case when the best migration interval is used in each processor.

The table indicates two interesting aspects about AM. First, fine-grained AM can achieve the frequency gain similar to that of coarse-grained AM. Especially, the frequency gain of PRO1 is completely equivalent to that of CLST. This is because CLST does not select the appropriate migration candidate, ifRAT which is the most dominant heat source in our base processor. In contrast, PRO1 correctly selects it, so it shows the significant frequency gain despite its area. Secondly, the frequency gain caused by AM increases as the temperature margin is enlarged. PRO1 achieves 15% of frequency improvement at 15 degree Celsius margin, while it achieves 10% of improvement at 10 degree Celsius margin. This is because static power depends on the chip temperature. Since the contribution of dynamic power consumption to the chip temperature is relatively large at low temperature, AM produces the big frequency gain.

The frequency gain shown in Table 7 is a little excessive comparing to the available frequency in real CPUs. This is because actual programs used for thermal tests are more stressful than 456.hmmer. As a result, real CPUs produce much more headrooms (greater than 15 degree Celsius) for 456.hmmer even when running at their maximum frequency. In fact, an active core shows around 40 degree Celsius during turbo mode when 456.hmmer runs on the processor “/w

| Processor | CPU frequency (relative frequency) |
|-----------|-----------------------------------|
|           | 10 C margin | 15 C margin |
| BASE      | 6.2 GHz (x1) | 5.4 GHz (x1) |
| AM-D      | 6.2 GHz (x1) | 5.6 GHz (x1.04) |
| CLST      | 6.8 GHz (x1.10) | 6.2 GHz (x1.15) |
| PRO1      | 6.8 GHz (x1.10) | 6.2 GHz (x1.15) |
| PRO2      | 6.8 GHz (x1.10) | 6.0 GHz (x1.11) |

Table 8 peak temperature (Celsius degree) for each unit when 456.hmmer runs at 5.4 GHz. An underlined value is the maximum number in each processor.

| Unit          | BASE | AM-D | CLST | PRO1 | PRO2 |
|---------------|------|------|------|------|------|
| L2,i          | 62.55 | 62.33 | 53.09 | 61.70 | 50.54 |
| FPU           | 64.70 | 59.22 | 56.51 | 64.49 | 59.83 |
| intrREG       | 71.14 | 69.24 | 58.26 | 68.46 | 62.62 |
| fpREG         | 62.92 | 55.99 | 56.92 | 62.24 | 55.37 |
| D-Cache       | 65.12 | 64.50 | 58.31 | 61.46 | 62.72 |
| Mult          | 64.03 | 64.92 | 62.07 | 63.44 | 64.44 |
| ahu,0         | 69.68 | 67.66 | 60.16 | 68.92 | 65.40 |
| ahu,1         | 69.48 | 67.60 | 60.77 | 68.74 | 65.40 |
| ahu,2         | 67.66 | 66.96 | 60.97 | 66.97 | 64.70 |
| ahu,3         | 65.83 | 65.87 | 61.18 | 65.18 | 64.54 |
| f-Cache       | 67.02 | 65.96 | 61.04 | 64.05 | 54.80 |
| BTB           | 71.83 | 70.77 | 57.54 | 67.13 | 56.70 |
| BPref         | 71.83 | 70.77 | 60.54 | 68.35 | 62.66 |
| Decoder       | 61.71 | 61.71 | 60.54 | 59.57 | 49.67 |
| LdQ           | 68.80 | 67.03 | 65.77 | 68.10 | 62.14 |
| StQ           | 68.10 | 66.45 | 62.27 | 65.45 | 56.58 |
| IntQ          | 69.47 | 67.80 | 53.82 | 65.96 | 57.27 |
| FpQ           | 62.57 | 55.41 | 57.44 | 61.69 | 54.67 |
| ROB           | 69.05 | 67.14 | 54.99 | 65.73 | 57.11 |
| I-TLB         | 61.31 | 61.27 | 53.16 | 59.50 | 52.29 |
| D-TLB         | 59.59 | 59.23 | 61.62 | 58.45 | 53.23 |
| fRAT          | 73.55 | 72.44 | 69.59 | 68.33 | 55.81 |
| iRAT          | 71.73 | 70.72 | 68.04 | 66.97 | 56.51 |
| iFree         | 62.55 | 62.53 | 64.38 | 61.70 | 50.06 |
| fFree         | 61.78 | 61.72 | 63.35 | 59.21 | 49.69 |
| iFree         | 62.55 | 62.53 | 62.55 | 61.70 | 50.06 |
| FPU,i         | -     | 54.21 | 59.21 | -     | -     |
| intrREG,i     | -     | 64.86 | 63.37 | -     | -     |
| fpREG,i       | -     | 52.37 | 64.08 | -     | -     |
| Mult,i        | -     | 64.84 | 63.55 | -     | -     |
| ahu,0,i       | -     | 66.69 | 63.78 | 65.86 | -     |
| ahu,1,i       | -     | 66.72 | 63.78 | 65.88 | -     |
| ahu,2,i       | -     | 66.43 | 63.35 | -     | -     |
| ahu,3,i       | -     | 65.55 | 62.81 | -     | -     |
| LdQ,i         | -     | -     | 66.73 | -     | -     |
| StQ,i         | -     | -     | 66.73 | -     | -     |
| IntQ,i        | -     | -     | 66.73 | -     | -     |
| FpQ,i         | -     | -     | 65.56 | -     | -     |
| D-TLB,i       | -     | -     | 62.31 | -     | -     |
| D-Cache,i     | -     | -     | 58.96 | -     | -     |
| iRAT,i        | -     | -     | -     | 69.18 | 62.62 |

The table also indicates that coarse-grained AM is not effective for reduction of the peak chip temperature. AM-D hardly lowers the peak temperature of the processor chip. This is because the hottest unit in BASE, ifRAT, is not dupli-
cated in AM-D. Although CLST also has no copy of ifRAT, it can reduce the peak temperature of ifRAT by 3.96 degree Celsius. This is because some cool units like LdQ (Load Queue) are installed in the adjacent area (Fig. 15).

In contrast, fine-grained AM effectively reduces the peak temperature. In PRO1, the hottest unit shifts from ifRAT to alu_1 by duplicating ifRAT. In addition, PRO2 achieves the further temperature reduction because of the duplication of alu_0 and alu_1. The peak temperature reduction in PRO1 and PRO2 is 4.63 and 7.67 degree Celsius, respectively.

Figures 17 and 18 present throughput (instruction per second) of five processors when they run at their maximum available frequency. The upper graph presents processors designed with 10 degree Celsius margin, while the lower graph presents processors designed with 15 degree Celsius margin. The horizontal axis represents programs, while the vertical axis represents throughput. The five bars indicate throughput of BASE, AM-D, CLST, PRO1 and PRO2, respectively. The throughput of each processor is normalized to that of the base processor. Only the best case for three migration intervals in each processor is shown in the figure.

The figure shows that fine-grained AM achieves high performance comparing to coarse-grained AM. AM-D and CLST show less performance improvement comparing to the base processor, while PRO1 and PRO2 show significant performance improvement. The throughput improvement in AM-D, CLST, PRO1 and PRO2 is −0.17%, 4.17%, 9.98% and 6.02% on average at 15 degree Celsius margin, respectively. Their improvement at 10 degree Celsius margin is −0.29%, 0.67%, 6.11% and 4.73%, respectively. The figure also indicates that turbo mode is strongly reinforced when CPU-bound workloads run. For example, PRO1 shows 14.5% of performance improvement for 456.hmmer at 15 degree Celsius margin. This is because CPU-bound workloads largely benefit from use of higher clock frequency comparing to memory-bound workloads like 429.mcf.

Figures 19 and 20 show IPC of the processors at 10 and 15 degree Celsius margin, respectively. The graphs present that CLST causes the significant IPC degradation. The IPC degradation is up to 18.1% (433.milc) at 15 degree Celsius margin. Since CLST presents 9.28% of IPC degradation on average, the throughput improvement is not so high despite its high available frequency. In contrast, fine-grained AM shows small IPC degradation comparing to CLST. The IPC degradation in PRO1 and PRO2 is 4.21% and 4.59% on average at 15 degree Celsius margin, respectively. This is the main reason that fine-grained AM shows large improvement in throughput.

Table 9 presents the average throughput improvement for four AM schemes when migration intervals change. The rows represent AM schemes, while the columns represent migration intervals. Generally speaking, more frequent migration causes larger reduction of peak chip temperature, so it achieves higher throughput improvement if its migration overhead is negligible. The results of PRO1 well express this trend. On the other hand, CLST does not match
Fig. 19  IPC of processors designed with 10 degree Celsius margin.

Fig. 20  IPC of processors designed with 15 degree Celsius margin.

Table 10  Area of processors.

| Name   | Area (mm²) | Relative area |
|--------|------------|---------------|
| BASE1&2| 30.481     | 1             |
| AM-D   | 36.400     | 1.194         |
| CLST   | 39.413     | 1.293         |
| PRO1   | 31.327     | 1.028         |
| PRO2   | 31.956     | 1.048         |

this trend because flushing the L1 data cache largely damages IPC. Therefore, infrequent migration is appropriate for CLST. AM-D does not show any changes against migration intervals because register files and execution units are not hottest in BASE. Therefore, changing migration intervals does not impact on the processor performance at all.

Table 10 summarizes the chip area of the processors. As shown in the table, coarse-grained AM largely increases the chip area. This is because the non-heat-source units such as multiplier, intREG, FPU and D-Cache (see Fig. 11) are doubled in coarse-grained AM. These units are huge, so duplicating these units significantly wastes transistors on processor chips. In contrast, fine-grained AM can lower the chip size as expected. The area increase of PRO1 and PRO2 is 2.8% and 4.8%, respectively.

According to Pollack’s rule, performance improvement is roughly proportional to the square root of area increase. That is, 2.8% of area increase causes around 1.67% of performance improvement according to this rule. Fine-grained AM can show the performance improvement beyond this rule. PRO1 achieves 9.98% of performance improvement (at 15 degree Celsius margin) in exchange for 2.8% of area increase.

From the overall results, we can conclude that fine-grained AM is a promising approach to achieve high performance and area-efficiency in microprocessors.

8. Related Work

Similar to ours, Karpuzcu et al. try to accelerate applications by focusing on thermal constraint [15]. They propose adding many cores into a processor chip and using them for acceleration in a disposable manner. During acceleration, only one core in core pools is activated and used with extreme over-clocking until it is broken. This approach can be regarded as a kind of inter-core AM. Although many applications can be a target of acceleration as well as us, an explicit drawback of this approach is to damage lifetime of hardware.

Although our approach is a kind of thermal management, its purpose is different from that of conventional DTM. Conventional DTM aims to prevent microprocessors from overheating, and therefore starts to work when chip temperature reaches a dangerous level caused by turbo mode [2]–[4], [28], [30]. In contrast, our purpose is to keep the peak chip temperature a lower level, and our method therefore always works during turbo mode.

Some researches of conventional DTM also use duplicating hot units for cooling, but it has been carried out in case-study methods. Skadron et al. propose duplicating an integer register file based on thermal simulation result [19]. Chaparro et al. propose cluster hopping that periodically switches clusters used for computation on clustered microprocessors [7], [8]. In contrast to these work, we decide duplicated units and their copy count based on the thermal model.
Except for AM, a few researches of preventive thermal management have been done. Bailis et al., present that periodic injection of idle cycles achieves high performance comparing to DTM with DVFS [5]. Powell et al. propose evenly using internal logic of some units such as ALUs, register files and instruction queues for temperature reduction [31]. Although these techniques do not require enlargement of processor chips, their effect of temperature reduction is limited.

Since our approach reduces power density of microprocessors in exchange for additional hardware, it can be regarded as a method to alleviate dark silicon problem where the advance of technology scaling yields a lot of inactive transistors [13]–[18]. General approaches for dark silicon problem are to use excess transistors as accelerators for specific codes [14], [16]–[18]. These approaches achieve higher performance under power constraint, but programs benefited from them are limited.

9. Conclusion

This paper first presented unit-level AM and the design methodology of processors with it. In contrast to coarse-grained AM, unit-level AM enables processors to avoid duplicating units unnecessary for temperature reduction. Since some limited units are heat sources in processors, unit-level AM can achieve high performance while lowering area increase. The paper also showed that the proposed design method was able to detect heat sources from base processors successfully and decide appropriate duplication count for unit-level AM. The effectiveness of our proposal was presented through the first thorough experiment of AM with various spatial granularity.

Temperature problems in microprocessors become more serious as LSI technology scaling advances, because power density of each unit is increased by shrinking die size. Thinning processor chips makes heat difficult to spread within silicon die. Therefore, preventive thermal management like unit-level AM will play an important role in the future microprocessors.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Number 24700044.

References

[1] Intel Corp., “http://www.intel.com/content/www/us/en/architecture-and-technology/turbo-boost/turbo-boost-technology.html?wapkw=turbo-boost”
[2] R. Cochran and S. Reda, “Consistent runtime thermal prediction and control through workload phase detection,” Proc. 47th Design Automation Conference, pp.62–67, 2010.
[3] C. Poirier, R. McGowen, C. Bostak, and S. Nafziger, “Power and temperature control on a 90nm Itanium-family processor,” Proc. 2005 IEEE International Solid-State Circuit Conference, pp.304–305, 2005.
[4] I. Yeo, C.C. Liu, and E.J. Kim, “Predictive dynamic thermal management for multicore systems,” Proc. 45th Design Automation Conference, pp.734–739, 2008.
[5] P. Bailis, V.J. Reddi, S. Gandhi, D. Brooks, and M. Seltzer, “Dimetrodon: Processor-level preventive thermal management via idle cycle injection,” Proc. 48th Design Automation Conference, pp.89–94, 2011.
[6] S. Heo, K. Barr, and K. Asanović, “Reducing power density through activity migration,” Proc. 2003 International Symposium on Low Power Electronics and Design, pp.217–222, 2003.
[7] P. Chaparro, J. Gonzalez, and A. Gonzalez, “Thermal-aware clustered microarchitectures,” Proc. IEEE International Conference on Computer Design, pp.48–53, 2004.
[8] P. Chaparro, J. Gonzalez, and A. Gonzalez, “Thermal-effective clustered microarchitectures,” Proc. 1st Workshop on Thermal Aware Computer Systems, 2004.
[9] S. Palacharla, N.P. Jouppi, and J.E. Smith, “Complexity-effective superscalar processors,” Proc. 24th Annual International Symposium on Computer Architecture, pp.206–218, 1997.
[10] M. Funck and R. Peterson, “Performance implications of POWER7 model 780’s TurboCore mode (white paper).”
[11] Hewlett-Packard Corp., Intel Corp., Microsoft Corp., Phoenix Technologies Ltd., and Toshiba Corp., “Advanced configuration and power interface specification rev. 4.0,” 2010.
[12] i7z project, ”https://code.google.com/p/i7z/”
[13] H. Esmaeilzadeh, E. Blem, R.S. Amant, K. Sankaralingam, and D. Burger, “Dark silicon and the end of multicore scaling,” Proc. 38th Annual International Symposium on Computer Architecture, pp.365–376, 2011.
[14] N. Goulding-Hotta, J. Sampson, Q. Zheng, V. Bhatt, S. Swanson, and M. Taylor, “GreenDroid: An architecture for the dark silicon age,” Proc. 17th Asia and South Pacific Design Automation Conference, pp.100–105, 2012.
[15] U.R. Karpuzcu, B. Greskamp, and J. Torrellas, “The BubbleWrap many-core: Popping cores for sequential acceleration,” Proc. 42nd Annual IEEE/ACM International Symposium on Microarchitecture, pp.447–458, 2009.
[16] M.B. Taylor, “Is dark silicon useful? Harnessing the four horsemen of the coming dark silicon apocalypse,” Proc. 49th Design Automation Conference, pp.1131–1136, 2012.
[17] G. Venkatesh, J. Sampson, N. Goulding, S. Garcia, V. Bryksin, J. Lugo-Martinez, S. Swanson, and M.B. Taylor, “Conservation cores: Reducing the energy of mature computations,” Proc. 15th International Conference on Architectural Support for Programming Languages and Operating Systems, pp.205–218, 2010.
[18] G. Venkatesh, J. Sampson, N. Goulding, S.K. Venkata, M.B. Taylor, and S. Swanson, “QcCores: Configurable co-processors to trade dark silicon for energy efficiency in a scalable manner,” Proc. 44th International Symposium on Microarchitecture, pp.163–174, 2011.
[19] K. Skadron, M.R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan, “Temperature-aware microarchitecture,” Proc. 30th Annual International Symposium on Computer Architecture, pp.2–13, 2003.
[20] N. Binkert, B. Beckmann, G. Black, S.K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D.R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M.D. Hill, and D.A. Wood, “The gem5 simulator,” SIGARCH Computer Architecture News, vol.39, no.2, pp.1–7, 2011.
[21] A. Patel, F. Afram, S. Chen, and K. Ghose, “MARSS: A full system simulator for multicore x86 CPUs,” Proc. 48th Design Automation Conference, pp.1050–1055, 2011.
[22] Processor Simulator Onikiri 2, “http://www.ml.t.u-tokyo.ac.jp/~onikiri2/”
[23] S. Li, J.H. Ahn, R.D. Strong, J.B. Blockman, D.M. Tullsen, and N.P. Jouppi, “McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures,” Proc. 42nd Annual IEEE/ACM International Symposium on Microarchitecture,
pp.469–480, 2009.

[24] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M.R. Stan, “HotSpot: A compact thermal modeling method for CMOS VLSI systems,” IEEE Trans. Compon. Packag. Manuf. Technol., vol.14, pp.501–513, 2006.

[25] W. Huang, S. Ghosh, K. Sankaranarayanan, K. Skadron, R.J. Ribando, and M.R. Stan, “Accurate, pre-RTL temperature-aware design using a parameterized, geometric thermal model,” IEEE Trans. Comput., vol.57, no.9, pp.1277–1288, 2006.

[26] Standard Performance Evaluation Corporation, “http://www.spec.org/cpu2006”

[27] J.L. Hennessy and D.A. Patterson, Computer Architecture: A Quantitative Approach, 5 ed., Morgan Kaufmann, 2011.

[28] Intel Corp., “CPU monitoring with DTS/PECI (white paper),” 2010.

[29] SimPoint, “http://cseweb.ucsd.edu/~calder/simpoint/”

[30] Y. Ge, P. Malani, and Q. Qiu, “Distributed task migration for thermal management in many-core systems,” Proc. 47th Design Automation Conference, pp.579–584, 2010.

[31] M.D. Powell, E. Schuchman, and T.N. Vijaykumar, “Balancing resource utilization to mitigate power density in processor pipelines,” Proc. 38th Annual IEEE/ACM International Symposium on Microarchitecture, pp.294–304, 2005.

Shinobu Miwa was born in 1977 and received the Doctor of Informatics degree from Kyoto University in 2007. He is now an Assistant Professor at the University of Tokyo. His research interests are computer architecture, high performance computing and embedded systems. He received the best paper award in 2010 Embedded System Symposium. He is a member of IEEE, IEICE and IPSJ.

Takara Inoue was born in 1988 and received B.E. from Tokyo University of Agriculture and Technology in 2011, and the Master of Information Science and Technology from the University of Tokyo in 2013, respectively. He currently works at Ryoyu Systems Co. Ltd. He is interested in computer architecture.

Hiroshi Nakamura is a professor in the Department of Information Physics and Computing at the University of Tokyo. His research interests include low-power VLSI design, power-aware computing, high-performance computer systems and dependable computing. He has a Ph.D. in electrical engineering from the University of Tokyo. He is a member of IEEE, the ACM, and IPSJ.