2009 International Conference on Computer Engineering and Applications
IPCSIT vol.2 (2011) © (2011) IACSIT Press, Singapore

Heat Spreading Aware Floorplan for Chip Multicore Processor[1]

Liqiang He and Cha Narisu

College of Computer Science, Inner Mongolia University, Hohhot, P. R. China

Abstract. With more cores integrated into one chip and multiple threads running concurrently on the chip, power consumption from the running threads increases dramatically, and then causes the thermal of the chip going much higher than before. Existed schemes to leverage the thermal problem for a chip multicore processor include DVFS, Clock Gating, thread migration, and so on. In this paper, we propose a new architectural method to address the thermal problem from another perspective, the floorplan of the multicore chip. Our method exploits the heat spreading model of a chip and tries to come up with a floorplan for a multicore processor which leaves the potential hotspots in different cores as far away from each other as possible. As an example, we show heat spreading aware floorplans for a four cores' processor and a sixteen cores' processor respectively. Coupling with the proposed floorplanning scheme, we propose a thermal aware thread mapping policy for the multicore processor. Trace-driven experiment results show that a processor with our floorplanning scheme always has lower/not higher peak temperature than the baseline processor, and further a processor using our thread mapping policy can have very good average and peak temperature among all the tested mapping schemes.

Keywords: Chip Multicore Processor, thermal, floorplan

1. Introduction

Chip multicore processor (CMP) has become the dominate architecture in computer industry and academic research literature. Quad core processor from Intel has been widely used in normal life, and the processor with eight cores and sixteen cores can be seen in not far away future.

With more cores integrated into one single chip, more and more threads can run concurrently on the chip. The parallel multiple threads running can dramatically increase the overall throughput of the processor through exploiting the different levels of parallelism. Whereas at the same time, power consumption from the running threads also increases largely which causes the chip temperature goes much higher than that in a traditional superscalar single core processor.

A chip multicore processor can have instantaneous hardware errors if the temperature surpasses the thermal limitation of the processor, or even broken if the temperature continuously goes up. In industry and academic research literature, some techniques are proposed and used to leverage the thermal problem of a chip multicore processor. Typically, Dynamic Voltage Frequency Scaling (DVFS) [1] decreases the supply voltage and operating frequency of a chip when the detected temperature is higher than the pre-defined thermal threshold, and after a period of interval when the temperature returns back to the safe region then increases the voltage and frequency step by step. Clock Gating [2] is to turn off the hot cores through turning off the clock thus that the static leakage power and dynamic power can be removed totally from the cores, and causes the temperature dropping down. Thread migration [3] is another scheme to leverage the thermal problem which tries to distribute the power dissipation from the running threads over the whole chip through migrating them over the chip. Although these techniques can relieve the thermal problem of a chip multicore processor, they also decrease the overall throughput of the processor or the performances of some specific programs when exploiting them.

[1] This work is supported by Inner Mongolia Natural Science Foundation Project No. 208091.

In this paper, we propose another orthogonal method which can help to address the thermal issue of a chip multicore processor. Our scheme is based on the observation that heat is another shared resource in a multicore processor. Each running thread contributes its own heat resource to the whole chip, and the heat will dissipate to the other cores and cause their temperature going up. On the other side, no matter whether a core has a thread running on it or not it can consume or absorb some heat resource from other cores and help to decrease their temperatures as a result. Under these two opposite effects, a chip multicore processor shows its final temperature to the out world. Using this heat dissipating model, our method tries to come up with a floorplan for a chip multicore processor which keeps the potential hot spots in different cores as far away as possible thus that maintains a longer heat dissipating distance among the hot spots in order to obtain a lower average temperature or a peak temperature for the processor. Figure 1 shows the rationale of this idea. Four possible floorplans for a two cores chip multicore processor are shown in Figure 1. Each core has an Alpha EV6 [4] structure. Suppose the hot spots occur at the red circular area. Floorplan A is a common floorplan in nowadays 2 cores processors, such as Intel Core Duo and AMD Dual-core. Among them, floorplan D has the longest distance for the potential hot spots such that the heat generated by two running threads in the cores can be more easily distributed on the chip and thus should has lower average temperature than the others. When running a same two threads workload in these two processors at same conditions, processor with floorplan D has lowest average temperature than the compared others because of the smallest hot area.

Due to hardware implementation complexity, we keep our method in the core granularity and do not go more deeply into functional unit granularity. But we believe this long heat spreading distance principal should always be considered when floorplanning for multicore processor, single core processor, and embedded processor if thermal issue is a key parameter for the processor. Another reason we do not consider this in functional unit level is because that different programs have different thermal behaviors, and too fine granularity may not fit different thermal characteristics of different programs.

Coupling with our floorplanning method, we also propose a thermal aware thread mapping policy for chip multicore processor. The basic idea is to try to benefit most from our proposed floorplan and map running threads to different cores according to their thermal characteristics such that the hot spots can be kept as far away as possible and the average or peak temperature of the chip can be lower.

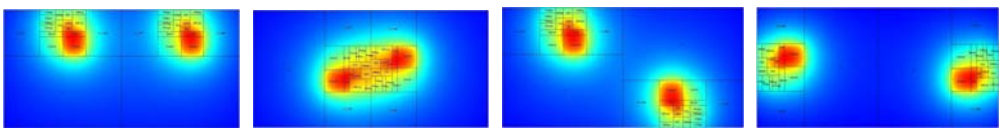
The main contributions of this paper are:

a. Showing that considering and analyzing heat spreading behaviours during processor floorplanning process is very important to the final run-time temperature of a chip multicore processor.

b. Propose a heat spreading model based floorplan scheme for chip multicore processor. With the proposed floorplan, the average and peak temperature is lower or at least not higher than the other floorplans at most of the time if without any thermal aware thread mapping policies.

c. Propose a thermal aware thread mapping policy which benefits from the proposed floorplan method and can obtain a reasonable better average and peak temperature than the other tested mapping schemes.

d. As an example, conduct simulation experiment for four cores and sixteen cores processor, and compare the proposed floorplan method with other floorplans. Experiment results show that in a four cores processor (a) one thread workload has same temperature as that in other floorplans, (b) two and three threads workload always have lower temperature in our floorplan, and (c) four threads workload has lower average temperature if mapped using our mapping policy. And similar results can also be observed in sixteen cores experiments.



(a) (b) (c) (d)

Figure 1. Steady-state temperatures of two gzips running in a two cores processor with four different floorplans.

The other parts of this paper is organized as follows: Section 2 presents our heat spreading model based floorplanning method, Section 3 gives our thermal aware thread scheduling policy, Section 4 describes experiment methodology and results, Section 5 discusses the related work, and Section 6 concludes this paper.

2. Heat Spreading Model Based Floorplan

2.1. Thermal Characteristics of Programs

In this subsection, we use SPEC CPU2000 benchmarks [5] as our workload to show the thermal characteristics of different programs. Experiments are done through a cycle-by-cycle accurate simulator, sim-alpha [6]. We add power model [7] and thermal model [8] in this simulator, and run 26 programs from the benchmark suite for 0.07s in a single core processor after fast forwarding different number of instructions using the technique similar as [9]. We collect the instantaneous temperatures of each function unit during the running period. The details of the simulated single core processor can be seen in Table 1.

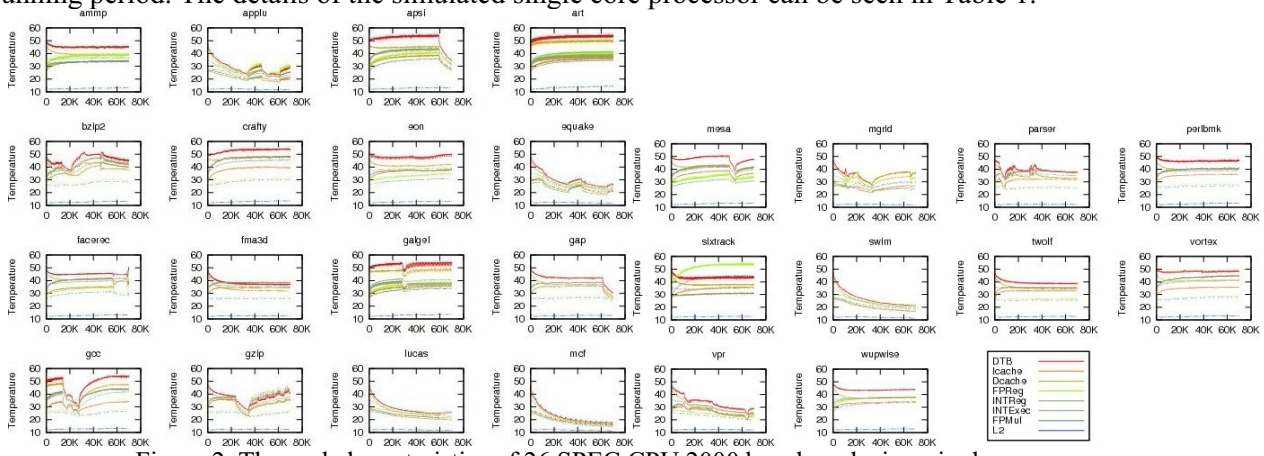


Figure 2. Thermal characteristics of 26 SPEC CPU 2000 benchmarks in a single core processor.

Figure 2 shows the different thermal behaviors of 26 SPEC CPU 2000 benchmarks during the running period (the sample interval is 0.001ms). The thermal values are relative temperatures rise above the ambient temperature (35 ^{o}C in our experiments). It can be seen that eight out of twenty-six programs (*applu*, *bzip2*, *equake*, *facerec*, *gcc*, *gzip*, *mgrid*, and *parser*) have very irregular thermal behaviors during the short running period and on the contrary other programs have relative regular ones. Thus the programs with irregular behavior call for finer granularity thermal management in order to avoid surpass the thermal threshold. For the regular ones, ten programs (*ammp*, *art*, *crafty*, *eon*, *fma3d*, *perlbmk*, *sixtrack*, *twolf*, *vortex*, and *wupwise*) are hot threads and have relative high average and peak temperatures that need to be scheduled or mapped smartly if they are running concurrently in a chip multicore processor in order to avoid the potential hotspots. And other four programs (*lucas*, *mcf*, *swim*, and *vpr*) are cold threads and can be used to absorb heat flux from co-running hot threads. Other four programs (*apsi*, *galgel*, *gap*, and *mesa*) change the thermal faces dramatically during the running period that also need a smart scheduling or mapping scheme.

When considering the different units in a core, L2 cache always has the lowest temperature. This is due to L2 cache has a big area on the chip, and the heat generated by L2 itself and the heat coming from other functional units can be dissipated easily into the air and thus causing the temperature dropping down. Similarly, Icache, Dcache, DTB, register file, and execution units have high temperatures due to high access frequencies and small areas.

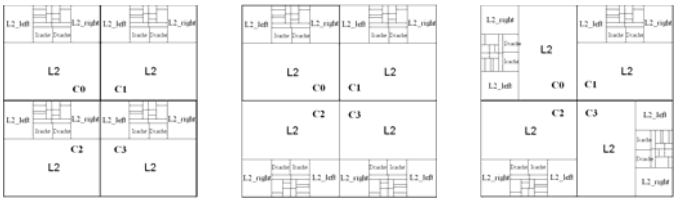
2.2. Relationship between Floorplan and Temperature

Most of the existed thermal aware floorplanning methods only work in a single core processor, such as [10, 11]. They try to find good block level placement for all the units in a processor in order to decrease the peak temperature. Few researches have done on a core level for a chip multicore processor. [12] adjusts the placement of multiple cores for performance gain but not for temperature. Our work in this paper is to do thermal aware floorplanning in core level for a CMP processor. Thus this section will present the relationship between different floorplans and the corresponding temperatures.

As shown in Figure 1, different floorplans in a two cores processor have different average and peak temperatures due to the heat spreading among the two cores. The longer is the distance between the hot spots in two cores, the lower is the temperature. The difference of the steady state temperature can be as high as 12 ^{o}C and 3 ^{o}C for peak and average temperature respectively in the experiment. Similar results can be seen in a four or sixteen cores' processor. These results show the importance of floorplanning in core level for a CMP processor.

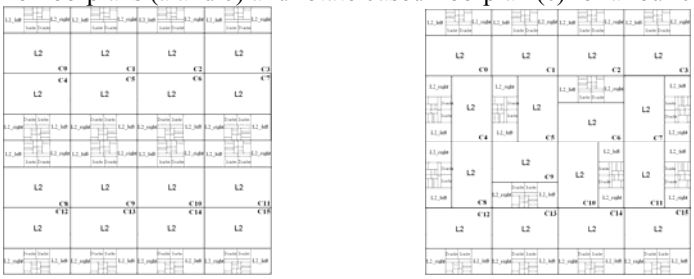
The rationale behind the experiment result is similar as that in previous researches for single core processor. For a single core processor, the maximum temperature of a block depends not only on its own power density (power consumption over the area) but also on the power density of the adjacent blocks.

Similarly, the maximum temperature of a core in a multicore processor depends not only on its own temperature, but also on its adjacent cores. So leaving potential hot spots in different cores as far away as possible is better for the temperature over the whole chip.



(a) (b) (c)

Figure 3. Baseline floorplans (a and b) and rotate based floorplan (c) for a four cores' processor.



(a) (b)

Figure 4. Baseline floorplan (a) and rotate based floorplan (b) for a sixteen core's processor.

2.3. Heat Spreading Aware Floorplanning

In this subsection, we use the analytical model in [8] as our heat spreading model to further understand the thermal behavior of a chip multicore processor. In [8], the power dissipated in transistors and wires is modeled by a prescribed heat flux *q(x, y, t)* depending on time *t* and representing the 2D power density in the plane, and for a specific point heat source the temperature on the plane is a function of the ***distance*** from the point position and as a function of ***time***. When multiple heat sources exist, the final temperature can be computed using space and time convolutions from all the heat sources. So for a microprocessor temperature model is the measurement of the effects of the different temperature sources on a chip over time.

In the following paper, we use the functional unit model instead of grid model in [8] to represent a processing core as well as a chip multicore processor in order to save simulating time, but this simplicity does not affect our conclusion because our method is orthogonal to theses models. In functional unit model, heat generated in a function unit is assumed to be distributed evenly over the whole area of the unit, and the temperature of the unit is measured at the centre point the unit.

Based on the thermal behaviors in subsection 2.1 and heat spreading model above, we propose a novel rotate based multicore processor floorplanning scheme. The basic idea is to keep potential hot spots (the function units that have high temperature, such as Icache, Dcache, DTB, register file, and execution units in Alpha EV6 core) in each core as far away as possible such that providing more opportunities to decrease the overall temperature of the whole chip. We do not consider the placement of blocks in a single core in this work because they have done by previous works. Also, we do not consider the interconnection of cores because it is out of the scope of this paper.

For a two core's processor, a rotate based floorplanning will give placement of cores as shown in Figure 2.d. Similarly, three possible floorplans for a four core's processor are shown in Figure 3, and our proposed floorplan is Figure 3.c. The difference between floorplan a and b is the placement of L2 cache. Floorplan b is reasonable in terms of implementation complexity. And the difference between floorplan b and c is the orientation of four separate cores. The rotation in floorplan c has more opportunities to smartly map threads into the cores in order to obtain a lower temperature due to the longer heat spreading distances.

Similarly for a sixteen cores' processor, there are many possible placements in core level. In Figure 4, we only show our baseline floorplan and our proposed rotate based floorplan. Our rotated floorplan tries to distribute the potential hot spots in the sixteen cores more evenly among the whole chip at the same time expose the hot regions of the cores at the edges close to the ambient environment. According to [13], this placement is better for the overall temperature.

For a CMP processor with more than sixteen cores, if the designers want to exploit our method they must consider different rotating possibilities and decide which one has the longest heat spreading distance among

the cores. The principle of design and selection will same as four cores and sixteen cores cases above. In this paper, we only use four cores and sixteen cores processor to explain our idea. We believe this idea should be considered in the design process of every chip multicore processor if temperature is a key issue for the processor.

3. Thermal Aware Thread Mapping Police

The temperature of a core in a CMP processor depends not only on the thread running on itself but also on the threads running on adjacent cores. To avoid hot spot due to inappropriate mapping two hot threads in two adjacent cores, a thermal aware thread mapping policy for CMP processor is needed. It is known that thread mapping/scheduling is a NP-Complete problem [14, 15]. In this section, we propose a simple greedy algorithm as shown in Figure 5 to get a sub-optimal placement for the running threads in a CMP processor. The complexity of our algorithm is O($n \times m$) where ***n*** is the number of cores and ***m*** is the number of threads being mapped.

The timing to make a new mapping using above algorithm has three options. One is the end of a fixed scheduling interval. Two is the time when a new thread comes in the running group. And third is the time when a thermal monitor detects a thermal face change of a running thread. The first two options do not need add any other support in existed hardware and software system. But the last option needs an extra thermal monitor mechanism in existed system.

[**Algorithm**]:

Let ***S*** is the set of current running tasks and ordered by the temperatures during last sample interval;
Let ***C*** is the set of available cores that can be mapped into tasks;
Let ***M*** is the set of pairs of *{Cores, Temperature}* that have been mapped into tasks;

C = {all the cores in a CMP processor};
M = NULL;
While (***S*** not Empty) do
    Take a task ***T*** which has the highest temperature in ***S***;
    WSP = -1; /* *WSP* is the weighted shortest path from core A to core B */
    C_s = -1;
    For all *c* in ***C*** do /* ordered by row major */

$$WP_c = \sum_{i=1}^{m \in M} Temperature_m \div D_{c\_m}$$

/* D_{c_m} is the distance between core *c* to core *m* */

        If (WP_c <= *WSP*) Then
            C_s = *c*;
            WSP = WP_c;
        End If
    End For
    Delete C_s from ***C***;
    Add *{Cs, Temperature$_c$}* to ***M***;
    Delete ***T*** from ***S***;
End While
/* Now M is the new mapping scheme */

Figure 5. Thermal aware thread mapping policy.

4. Methodology

We use sim-alpha to run 26 SPEC CPU programs and collect the instantaneous power density values during the predefined 0.07s running period. The parameters of single core are shown in Table 1. To build a chip multicore processor, we compose multiple same single cores to construct a multicore chip. Hence, our multicore chip is a homogeneous CMP. We do not consider heterogeneous CMP in this work. Also, we do not address the interconnecting issue among the cores at here.

Temperatures of multicore processor are simulated using the analytical model in [8] through feeding the power density values of each thread in a workload and a designed floorplan for the processor. We construct two, three, and four threads workloads as shown in Table 2 separately according to their different thermal behaviors for the simulated four core's CMP processor and compare the average and peak temperatures of the baseline floorplan and our proposed floorplan. The average and peak temperature are computed using the

following formula (1) and (2) and the T_{ij} is the temperature of function unit j in core i. Also, we do the similar experiments for a sixteen core's processor.

We compare the different mapping schemes in a sixteen core's processor. Hundred and ten different maps for the eight threads workload listed in Table 2 are generated using a random technique, and compare the average and peak temperatures with our proposed thermal aware mapping policy. Also, we construct one twelve threads workload and one sixteen threads workload that each workload includes the smaller one.

Table 1. Parameters of a single core in simulated CMP processor.

Parameter	Value
frequency	3GHz
fetch queue	16 entries
fetch/slot/map wid.	4 inst
issue width	4(int)/2(fp)
Reg	128(int,fp), 2 cycles lat.
ld/st queue	32/32 entries
functional units	4 INT ALU's, 4(INT mult/div), 1 FP ALU, 1 FP mult/div
Icache/Dcache/L2	64/64KB/2MB, 2/2/4 way, 64B block, LRU, 1/3/10 cycle lat.
memory latency	min 166, max 255 cycles
itlb/dtlb	2KB, 4way, 1 cycle lat.
pipeline depth	16
branch predictor	bimod/gshare comb. 16KB, 64RAS

$$Temp_{avg} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{k} T_{ij}}{n \times k} \quad (1) \qquad Temp_{peak} = MAX(T_{ij}) \quad (2)$$

Table 2. Workloads in the experiments (H: hot, M: middle, and C: cold).

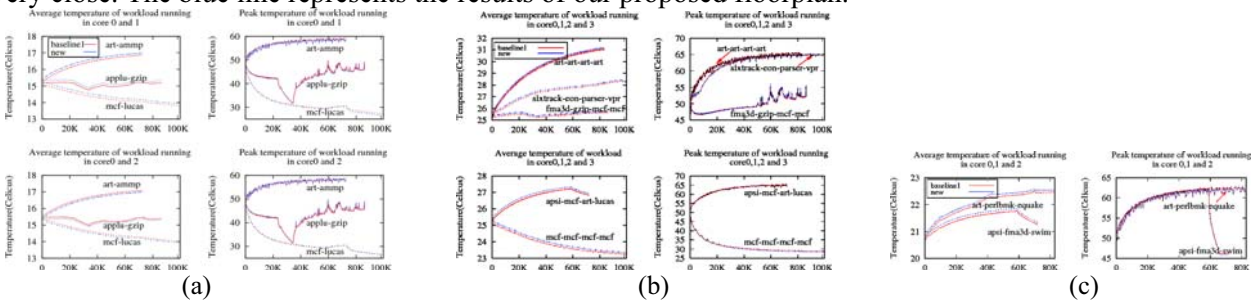
Workload	Type
art-ammp, applu-gzip, mcf-lucas	HH, MM, CC
apsi-fma3d-swim, art-perlbmk-equake	HMC, HHC
art-art-art-art, apsi-mesa-fma3d-art, apsi-mcf-art-lucas fma3d-gzip-mcf-mcf, sixtrack-eon-parser-vpr,mcf-mcf-mcf-mcf	HHHH, HHMH, HCHC MMCC, HHMC, CCCC
art-applu-crafty-facerec-fma3d-mesa-perlbmk-twolf (above)+galgel_vortex_swim_mcf (above)+eon+wupwise+gzip+gap	HLHMMHHM (above)HMCC (above)MMMM

5. Experiment Result

5.1. Four cores experiment result

(a) Two threads workload

Figure 6.a shows the relative average and peak temperatures (relative to ambient temperature, same in the following figures) of three two thread workloads running in a four core processor with different floorplans (Shown in Fig. 3). X axis is the number of sample points in time, and the sample interval is 0.01ms, same in the following figures. Also, we only show the result of one baseline because the values of the two ones are very close. The blue line represents the results of our proposed floorplan.



(a) (b) (c)

Figure 6. Average and peak temperatures of 2 (a), 4 (b)and 3 (c) threads' workloads.

When two threads are scheduled to run in core 0 and core 1, the peak temperatures of the three workloads with our floorplan are lower but close to the ones with the baseline floorplan. This is due to the little longer dissipation distance between the two threads in our floorplan. And because some of the heat is transferred to the other components in the same chip, the average temperature has a little increasing.

Similarly, when two threads are scheduled to run in core 0 and core 2, our floorplan provides even longer heat dissipating distance when compared with baseline floorplan and the above core 0 and core 1 mapping

scheme. So for the same three workloads, the peak temperature shows a little bit more decrease than the baseline and the previous result although the absolute value is still less than 1 ^{o}C. Further more, the average temperatures also decrease and are lower than the baseline.

When the threads are scheduled to run in core 0 and core 3, our floorplan has the same heat dissipating distance as the baseline thus the average and peak temperatures are totally same for the three workloads.

In summary, when the four cores processor has less work stress, our proposed floorplan always has lower or not higher peak temperature than the baseline, and when combining with a good mapping scheme it also has less average temperature for the processor.

Decreasing the peak temperature is the key objective of our floorplan because it can reduce the number of triggering of the DTM (Dynamic Thermal Management) techniques and thus decreasing the performance losing caused by the DTM. Our floorplan does not introduce any implementation complexity, just rotate and can has lower peak temperature than the baseline although the absolute value is not high.

(b) **Three threads workload**

Figure 6.c shows the three threads workload experiment results. In this case, we do not have other mapping possibility for our floorplan. Similar as in two threads experiments, our floorplan always has lower peak temperatures for the three workloads and a little bit higher average temperatures due to the heat dissipating to the other components in the same chip.

(c) **Four threads workload**

Figure 6.b shows the experiment results of four thread workloads. The trend is same as the two and three thread experiments. The peak temperatures of our floorplan are lower than the baseline and close to 1 ^{o}C decreasing for some high temperature stress workloads. And the average temperature is a little bit higher than baseline result due to the same reason as in previous experiments.

In summary, our heat spreading based floorplan can effectively decrease the peak temperature of the processor, and the absolute decreasing value goes up with the increasing of the number of concurrent running threads. And due to the heat dissipating to the other components in the same chip the average temperature increases slightly.

5.2. Sixteen cores experiment result

In this subsection, we show the effectiveness of our mapping policy. All the experiments are done using our proposed floorplan for sixteen cores' processor (shown in Figure 4.b). As known before, thread mapping in a multicore processor is a NP-complete problem. We exploit our greedy algorithm to map multiple threads in the processor. The inputs are the knowledge of the temperatures of the scheduled threads in previous running interval. To be able to have enough working stress, we construct one eight threads' workload, one twelve threads' workload, and one sixteen threads' workload as shown in Table 2. Temperature comparing is conducted between our mapping scheme and some other random generated mappings. The temperatures of threads selected to run in a workload in the previous interval is sorted and shown in Table 3.

Table 3. Temperatures (^{o}C) of threads in previous running interval.

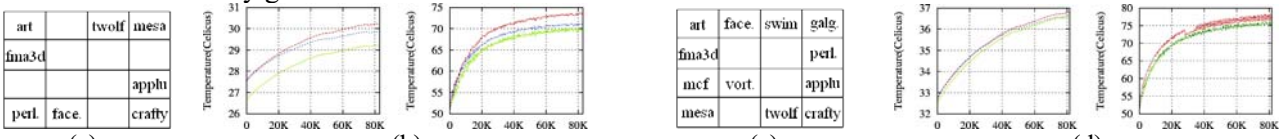
art:54	crafty:53	galgel:52	mesa:50
vortex:49	eon:48	perlbmk:47	facerec:46
gap:45	wupwise:44	gzip:41	twolf:40
fma3d:38	applu:35	swim:28	mcf:25

(a) Eight threads workload

Figure 7.a shows the mapping scheme for the eight threads' workload. It puts the hottest two threads, *art* and *crafty*, at the two corners of the chip, and then calculates the heat dissipating distances for each of the other threads. This mapping can potential leave the hot spots in a workload as far as possible. It is worth to mention that our scheme can not give the optimal mapping for the workload, but a sub-optimal solution with a polynomial complexity is acceptable for us in this work.

Figure 7.b shows the average (left) and peak (right) temperatures of the workload running with different mappings. For sixteen core processor, the different mappings for eight threads' workload can be up to p_{16}^{8}. It is impossible to investigate all the possibilities of mapping. So we randomly generate hundred and ten different mappings and compare the temperatures of them with the one of our mapping. In Figure 7.b, the red line and the green line show the upper bound and the lower bound of the temperature respectively, and the blue line shows the result of our mapping.

From Figure 7.b, the difference of all the average temperatures is about 1°C, but the difference of the peak temperatures can be up to 5°C for different mapping schemes. This shows that there is a big space to improve for the peak temperature from the mapping policy. The peak temperature of our mapping scheme is very close (1°C) to the lower bound because of the heat spreading model theory and shows the effectiveness of our scheme. For the average temperature, our mapping scheme has the same value as the upper bound at first but later continuously goes down and close to the lower bound.



(a) (b) (c) (d)

Figure 7. Thread mapping of workload and the average and peak temperatures.

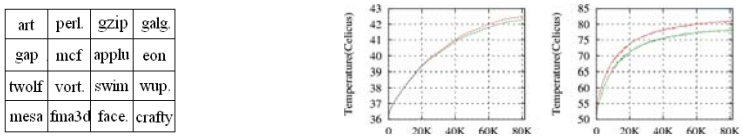
(b) Twelve threads' workload

Figure 7.c shows the mapping of our scheme for twelve threads' workload. Due to four new threads are added, the mapping is adjusted according to the new thermal characteristics. And Figure 7.d shows the average (left) and peak (right) temperatures of our scheme and other different mappings. Due to time limitation, we only generate twenty seven tested mappings in this experiment.

In Figure 7.d, the peak temperature of our scheme is very close to the lower bound temperature among all the tested mappings. And the average temperature of our scheme also goes from the upper bound temperature at first to the lower bound temperature at last. The absolute value of the difference between upper bound and lower bound is about 3 °C for peak temperature and 0.3 °C for average temperature. Again, due to the limited number of tested mappings these differences are not so big in this experiment, but with the number of different mappings increases we believe our scheme can show more benefit from other schemes.

(c) Sixteen threads' workload

Figure 8.a shows the mapping of our scheme for sixteen threads' workload. At this time, the processor has a fully utilization and a big thermal challenge if multiple hot threads concurrent running as shown in this experiment. We random generate other twenty-three mappings for this workload. The average and peak temperatures are shown in Figure 8.b. Similar as in twelve threads experiment, our mapping scheme gets the lowest average and peak temperature among all the experiments which shows the effectiveness of our algorithm again.



(a) (b)

Figure 8. The mapping (a) of twelve threads' workload and its average (b.left) and peak (b.right) temperatures.

In summary, our thermal aware thread mapping policy can effectively find a sub-optimal mapping scheme for a multiple threads workload and obtain a good average and peak temperature thus decrease the number of triggering DTM techniques and improve the final performance as a result.

6. Related Work

In academic research literature, numerable thermal-ware floorplan schemes have been proposed [10, 11] but only for traditional superscalar single core processor. The idea of these works is exploiting the thermal model to place blocks of a chip thus that the average/peak temperature can be keep low. Little work has been done for multicore processor. [12] proposes couple of thermal-ware floorplanning algorithms for a heterogeneous MPSoC architecture, whereas in this paper we focus on homogeneous multicore processor. [13] gives a rotate based multiprocessor organization which separates the last level cache into banks and surround the cache banks around the rotated cores. The objective of the work is to maximize cache hit rate but our work in this paper is to minimize processor temperature.

For thermal-ware task scheduling in multicore processor, there are also many works done by academic and industry researchers. [14] combines application power information and core's temperature information into together and make scheduling decision for new coming tasks. It does not take conjoint cores into consideration which limits the benefit of their algorithm. [15] considers the temperatures of both the scheduled cores and all the neighbor cores surround them and proposes a formula to compute the costs and selects the core with the lowest cost as the scheduled one. Unfortunately, the formula depends on five experiment detected parameters and these parameters may not suitable for different kind of applications.

Some other works [16] mostly focus how to select group of tasks to be concurrently running in a multicore processor and not consider how to map the tasks to different cores on the chip

7. Conclusion

Thermal-aware floorplanning has attracted many attentions from computer architects. Most of existed schemes focus on single core floorplanning method and ignore the potential benefit from thermal aware floorplanning in cores' level. In this work, we propose a heat dissipation model based floorplanning method for chip multicore processor. We limit our work on cores' granularity and show that there can be more space to leverage the thermal and power problem for a CMP processor through carefully floorplanning the on chip cores. Although the experiment results show little improvement of the peak and average temperature, but the decreased number of DTM triggers can be used to improve the final performance effectively. In addition, to get more benefit from our floorplanning scheme, we propose a thermal-aware thread scheduling algorithm. The idea of our algorithm is to keep the potential hot spots in different cores as far as possible thus to decrease the average and peak temperature of the whole chip. Although thread mapping is a NP problem, our algorithm can find a sub-optimal mapping solution with a polynomial complexity cost. Experiment results show that comparing with up to hundred of tested mappings our mapping's temperature is very close to lower bound of the range and it shows the effectiveness of our scheme.

In the future, we will conduct more experiments to test and verify the temperature benefit of our proposed floorplanning through making longer sample period and more different workload combinations. In addition, we will do more researches on thread scheduling method on a heterogeneous and homogeneous CMP processor and compare our scheme with other existed schemes in the literature.

8. Reference

[1] B. Zhai, D. Blaauw, D. Sylvester and K. Flaunter, Theoretical and Practical Limits of Dynamic Voltage Scaling, *DAC*, USA, pp.868-873, June 7-11, 2004

[2] F. Emnett and M. Biegel, Power Reduction Through RTL Clock Gating, *SNUG*, San Jose, 2000.

[3] S. Heo, K. Barr, and K. Asanovic. Reducing Power Density through Activity Migration. In *Proc. of the International Symposium on Low Power Electronics and Design*, pp. 217-222, Seoul, South Korea, 2003.

[4] E. J. McLellan and D. A. Webb. The Alpha 21264 Microprocessor Architecture. In *Proc. of the International Conference on Computer Design*, 1998.

[5] J. L. Henning, SPEC CPU2000: Measuring CPU Performance in the New Millennium, *Computer*, vol. 33, no. 7, pp. 28-35, July 2000

[6] R. Desikan, D. C. Burger, S.W. Keckler, and T. Austin. Sim-alpha: a Validated, Execution-Driven Alpha 21264 Simulator. The University of Texas at Austin, Department of Computer Sciences. Technical Report TR-01-23.

[7] W. Huang, S. Ghosh, K. Sankaranarayanan, K. Skadron, and M. R. Stan. HotSpot: Thermal Modeling for CMOS VLSI Systems. *IEEE Transactions on Component Packaging and Manufacturing Technology*. 2005

[8] P. Michaud and Y. Sazeides, A. Seznec, T. Constantinou, and D. Fetis. An Analytical Model of Temperature in Microprocessors. Research report RR-5744, INRIA, Nov. 2005

[9] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder, Automatically Characterizing Large Scale Program Behavior, *10th ASPLOS*, Oct. 2002.

[10] K. Skadron, A Case for Thermal-aware Floorplanning at the Microarchitectural Level, in *Journal of Instruction-Level. Parallelism*, 2005.

[11] W-L. Hung, Y. Xie, N. Vijaykrishnan, C. Addo-Quaye, T. Theocharides, M. J. Irwin. Thermal-Aware Floorplanning Using Genetic Algorithms. *Proc. of International Symposium on Quality Electronic Design*, pp. 634-639, Mar. 2005

[12] H. Mogal, and K. Bazargan, Thermal-Aware Floorplanning for Task Migration Enabled Active Sub-threshold Leakage Reduction, In *International Conference on Computer-Aided Design*, 2008

[13] Z. Guz, I. Keidar, A. Kolodny, and U. Weiser, Nahalal: Cache Organization for Chip Multiprocessors, *IEEE Computer Architecture Letters*, vol. 6, no.21-24, 2007

[14] Yeo, I., Liu, C. C., and Kim, E. J.. Predictive dynamic thermal management for multicore systems. In *Proc. of the 45th Annual Conference on Design Automation. DAC '08*. New York, 734-739.

[15] Stavrou, K. and Trancoso, P. 2007. Thermal-aware scheduling for future chip multiprocessors. *EURASIP J.* Embedded Syst. 2007, 1.

[16] Choi, J., Cher, C., Franke, H., Hamann, H., Weger, A., and Bose, Thermal-aware Task Scheduling at the System Software Level. In *Proc. of the 2007 international Symposium on Low Power Electronics and Design. ISLPED '07.* New York, 213-218.