Temperature Aware Thread Migration in 3D Architecture with Stacked DRAM

Dali Zhao[1], Houman Homayoun[2] and Alex V. Veidenbaum[1]
[1]University of California, Irvine
[2]George Mason University
daliz@uci.edu

***Abstract*—A 3D architecture with DRAM memory stacked on a multi-core processor has many benefits for the embedded system. Compared with a conventional 2D design, it reduces memory access latency, increases memory bandwidth and reduces energy consumption. However it poses a thermal challenge as the heat generated by the processor cannot dissipate efficiently through the DRAM memory layer. Due to the fact that DRAM is very sensitive to high temperature as well as temperature variance, 3D stacking causes more failures to occur because DRAM thermal variance is higher than the conventional 2D architecture. To address this thermal challenge we propose to reduce temperature variance and peak temperature of a 3D multi-core processor and stacked DRAM by thermally aware thread migration among processor cores. This method has very limited impact on processor performance. Using migration-based policy we reduce peak steady-state temperature in the processor by up to 8.3 degrees Celsius, with the average of 4.7 degrees.**

***Index Terms*—Thermal Management, 3D architecture**

I. Introduction

The 3D stacked architecture with DRAM memory on top of a processor provides a number of benefits for system performance, energy consumption, and packaging density compared to a conventional 2D design. This has been advocated for high-performance as well as embedded processor designs. In the latter case, in particular, the higher packaging density is a major advantage, in which a limited amount of DRAM in such systems allows a complete 3D SoC solution. It has already appeared in systems in the package-on-package form. For instance, the Apple's iPhone 4S is supposed to use the A5 processor, an SoC with two LPDDR2 SDRAM chips on top as a package-on-package type of design[1].

However, this type of DRAM stacking also creates new challenges. First, with one or more layers of memory stacked on top of a processor layer, the direct path of heat dissipation from the processor to the heat sink is interrupted, thus causing higher peak and steady-state temperatures in both the memory and the processor layer. High peak temperature complicates the packaging of the chip and increases costs [2]. It also reduces the reliability and wear out of both the processor and the DRAM [3], [4], [5], [6]. Second, the energy consumption of each processor core could vary widely due to different workloads. Also, different components within a processor core consume different amount of energy (e.g. a register file and an L2 cache). The resulting thermal variation, i.e. the difference in temperature among different locations in a processor, impacts the reliability of the DRAM and the processor. [3], [6]. We define the thermal variance between two components inside a processor as the difference in temperature between these two components. If the temperature within a component is not uniform, we define thermal variance as the difference in peak temperatures between these components, as peak temperature is more significant in thermal-aware designs.

This paper focuses on stacking one or more DRAM layers on top of a multi-core processor die. This type of design is more suited to the embedded domain where area and packaging are major constraints. The processor contains an L2 cache shared by all cores. A TSV stripe similar to [7] is assumed to connect the multiple layers. It is easy to see how this type of 3D stack will have higher temperatures at the processor layer as well as the DRAM layer.

The problem addressed by this paper is how to reduce both peak and steady-state temperatures in a 3D stacked DRAM/processor architecture. It proposes to use thread migration to address the above-mentioned power and thermal problems. The thread migration policy is shown to achieve a significant temperature reduction with little impact on performance. To be equally effective, other approaches such as DVFS or sleep modes cause a significant performance degradation, as temperature changes are quite slow. This paper examines a number of migration algorithms and their performance impact. It shows that a migration policy integrated with the OS context switching mechanism can reduce the peak and steady-state temperatures significantly. If thermal sensors on the processor cores are available, the migration algorithm can be improved to generate fewer OS context switches.

This paper aims to migrate a thread running on processor core C_i to another core C_j when C_i gets too hot. This is done periodically (10ms, which is equivalent to processor context switching time). This low migration frequency is chosen for the following reasons: a) the temperature rises or falls slowly, long after the power was increased or decreased. A fast migration frequency is therefore not necessary. b) thread migration negatively affects the performance due to context switch and cache state migration overheads. However the effect is relatively small.

The first migration algorithm rotates the threads in a round-robin fashion across all processor cores. It works well when the workloads and power dissipation are significantly different. This algorithm averages the power consumption of all processor cores and results in the least thermal variance. A major advantage of this algorithm is that it doesn't require thermal sensors to monitor core temperatures. However, this algorithm migrates all the threads across the cores all the time and thus has the highest performance overhead. Our evaluation shows that this algorithm reduces the peak temperature, on average,

by 4.16 °C and the thermal variance by 4.48 °C.

With thermal sensors, we can use an algorithm to only swap the hottest and the coldest cores. This algorithm reduces peak temperature by 3.86 °C. To further reduce the number of migrations, we migrate only when the thermal variance between two processor cores is larger than a threshold. Our experiment suggests that 2 °Cis a good choice for threshold. Using this threshold, we reduce the number of migrations by 43.3% on average, and reduces the peak temperature and variance by 3.72 °C and 1.39 °C, respectively.

The rest of the paper is organized as follows. Section II compares our approach with other related work. In section III, we present three migration algorithms and a method to dynamically reduce migrations that are unnecessary. Section IV describes our 3D architecture and the power and thermal model. It then explains the experiment methodology. V evaluates the algorithms with simulation results. Section VI and VII presents our future work and conclusion remarks.

II. Related Work

There is a large body of research regarding 3D architecture design. In the realm of processor design, thermal-aware floorplaning [8] tries to optimize the design of the processor to spread heat more easily. Thermal Herding [9] tries to move the circuitry that consumes more energy closer to the heat sink. Our approach is a software solution that adapts to different applications at run time, which is very difficult to achieve at the hardware design phase.

Single-thread activity migration [10] allocates a duplicate set of processor components so that when some components get hot, execution is shifted to the duplicate unit. This approach requires extra logic components that are on "stand by".

In the realm of run-time thermal management, for both 2D and 3D designs, researchers have proposed thermal aware job scheduling to reduce peak temperature [11], voltage and frequency scaling [12]. Coskun et al. has proposed a dynamic thermal management scheme for 3D multi-core processors [13]. They target a complicated 3D architecture where processors are stacked on top of each other. They proposed technique always assigns the new job to the coolest core to achieve thermal balancing across the 3D chip. [14] claims that the temperature of vertically adjacent cores has very strong correlation. Leveraging this observation, the authors proposed to wrapped up vertically stacked cores into super cores. Accordingly, tasks are also wrapped into super tasks. Then the hottest super job is assigned to the coolest super core in order to achieve the thermal balance. Thermal aware migration focuses on a less complicated architecture, which shows good potential in the embedded domain. Our approach offers a simple yet effective solution to the thermal challenge in 3D stacked DRAM architecture.

III. Thread Migration Algorithms

A. Motivation

Figure 2a shows the steady-state thermal map of a four-core processor running SPEC2000 benchmarks. On each processor core, the functional units in the center are much hotter than the private L2 cache surrounding it, with a thermal variance of at least 10 °C. The hottest functional unit in the bottom-left core, which is the floating point adder, is 30 °C hotter than the L2 cache. The peak temperatures for each core in degrees Celsius are $74.10, 64.22, 67.43$ and 55.81 respectively, starting from the bottom-left and going clockwise. The thermal variance between these cores can be as high as 18.29 °C. If we could migrate the threads between these different processor cores, we could fill the temperature gap with the temperature peak, thus reducing the thermal variance.

Figure 2b shows the thermal map of a 4Gb DRAM with 16 banks. Because the temperature is much lower, as well as the variance, we use a different thermal scale here. We can clearly see that the activity on the 2nd bank of the 1st row is higher than the others. However, the thermal variation among the banks is within 0.6 °C. From this, we can draw the conclusion that DRAM access pattern in the multi-core processor is uniformly distributed among different banks and the thermal variation within the memory is small. This kind of uniformly distributed access pattern is observed because the memory hierarchy is working well, which is the case for the majority of workloads. If we stack memory on top of the processor in a 3D architecture, the underlying processor itself and the interaction between layers will be the major causes of thermal problems for the DRAM. Therefore, it's important to reduce the peak temperature and variance in the logic layer. Prior proposals to reduce peak temperature usually involved aggressive action such as shutting down a processor core completely. This has undesirable effects on performance. Thread migration is a better way to control peak temperature and thermal variation as well because the effect on performance is small. Since the operating system constantly schedules threads, thermal-aware thread migration can be incorporated into the OS with little effort.

We propose four thread migration algorithms: *rotation, pairwise, dynamic14* and *dynamic23*. Rotation and pair-wise are static algorithms, which means they always migrate threads even the thermal variance is small. The other two dynamic algorithms attempt to reduce the number of unnecessary migrations by migrating only when the variance is large enough.

B. Rotation Algorithm

The first migration algorithm rotates the threads clockwise. We investigate this algorithm because of its simplicity. One biggest advantage of this algorithm is the fact that it does not require temperature sensors. As we observed in Figure 2a, sometimes the peak temperature only exists in a small area. Due to the limited placement of thermal sensors, we cannot always acquire the accurate reading of the peak temperature. Therefore, algorithms requiring thermal sensor is limited by thermal sensor range of error. Figure 1 (a) shows an example of the rotation algorithm. As time progresses, the threads running on each core are rotated clockwise. In this way every thread has an equal chance to execute on each processor core,

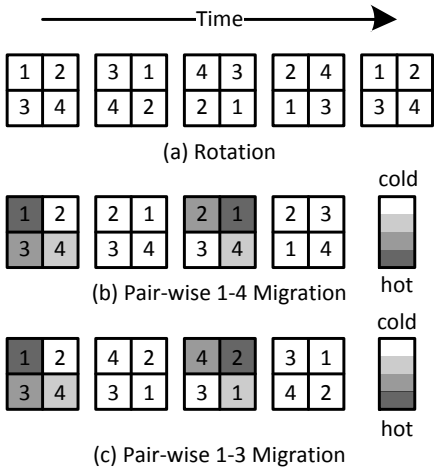


Fig. 1. Illustration of how threads migrate in the 4 core processor.

therefore the thermal variance and the peak temperature will be reduced.

The disadvantage of the rotation algorithm is its relatively high cost. Rotating all four threads is more costly than swapping pairs of threads because it forms a dependency chain and must be done in series. Swapping 2 pairs can be done in parallel.

C. Pair-wise Algorithm

Unlike the rotation algorithm, the pair-wise algorithm requires temperature sensors. At each time of migration, we sort the threads based on the temperature of the processor core that they run on, and name them thread 1, 2, 3 and 4. We then swap the two hot-cold pairs. There are 2 ways of selecting the pairs: $(1,4),(2,3)$ and $(1,3),(2,4)$. Figure 1 (b) and (c) show these two types of migrations. The number represents each running thread and the background color represents the processor core temperature. The second and fourth column shows the thread position right after the migration, with a white background.

For 1-4 migration, as described in algorithm 1 and figure 1 (b), the hottest and the coldest threads switch places and the other two threads also switch. As we can see in the figure, 1 is the hottest thread and 2 is the coldest. Therefore 1 and 2 switch places, also do 3 and 4. In the second round, 1 is still the hottest thread, but 3 become the coldest. So 1 and 3 switch, and so do 2 and 4. For 1-3 migration, the hottest and the second coldest switch places, and the second hottest and the coldest switch. As we see in figure 1 (c), 1 is the hottest thread and 4 is the second coldest, so they switch places. The other pair, which is the second hottest and the coldest, also switch.

Algorithm 1 PAIR-WISE 1-4 MIGRATION ALGORITHM

loop
  sleep for $Int_{thermal}$
  Sort Core ID by temperature as array $T[i]$
  Swap threads on $Core_{T[0]}$ and $Core_{T[3]}$
  Swap threads on $Core_{T[1]}$ and $Core_{T[2]}$
end loop

D. Dynamic Migration

We have observed that even with the most rigorous algorithm, *rotation*, thermal variance still exists, which raises the question whether all migrations are necessary. It turns out, not all migrations are necessary. Since there will be a limit on how much variation could be reduced, we don't have to migrate if the variance between the 2 treads is below a certain threshold T_th. As described in algorithm 2, at each migration interval, the decision whether or not to migrate is made dynamically and based on the thermal variation at that time. Using this dynamic method the unnecessary migrations could be reduced which in turn could reduce the impact on performance.

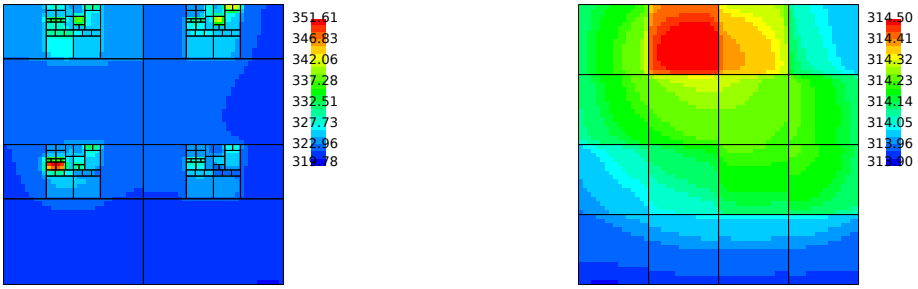
Algorithm 2 DYNAMIC MIGRATION ALGORITHM

1: **loop**
2: Sleep for $Int_{thermal}$
3: Sort Core ID by temperature as array $T[i]$
4: **if** $CoreTemp_{T[0]} - CoreTemp_{T[3]} \geq T_{th}$ **then**
5: Swap threads on $Core_{T[0]}$ and $Core_{T[3]}$
6: **end if**
7: **if** $CoreTemp_{T[1]} - CoreTemp_{T[2]} \geq T_{th}$ **then**
8: Swap threads on $Core_{T[1]}$ and $Core_{T[2]}$
9: **end if**
10: **end loop**

E. Cost of Migration

Since the change in temperature happens slowly, thread migration due to thermal concerns does not have to occur frequently. Also, the most convenient time to use this algorithm would be OS context switch. The OS only does little extra work, and the negative impact on performance is overlapped with the regular context switching. Therefore we choose the interval of $10ms$, which is the same as OS context switching time. Due to the similarity of context switch and tread migration, we use context switch cost to estimate the cost of thread migration.

Craeynest et al. studied the cost of context switching in [15]. In their model, two SPEC CPU2006 workloads are running on a dual-core processor with shared last level cache (LLC). They simulated all possible two-workload mixes of the benchmark suite with various context switching intervals. Their result shows that with the switching interval of $2.5ms$, the context switch overhead is within $\%0.4$. The low overhead can be explained by the coherence mechanism and the shared LLC. A migrating thread looses its L1 cache state and increases L1



(a) 4-core Processor (b) DRAM

Fig. 2. Thermal map of processor core and DRAM without stacking. Temperature is in Kelvin. Different temperature scales are used due to high temperature difference.

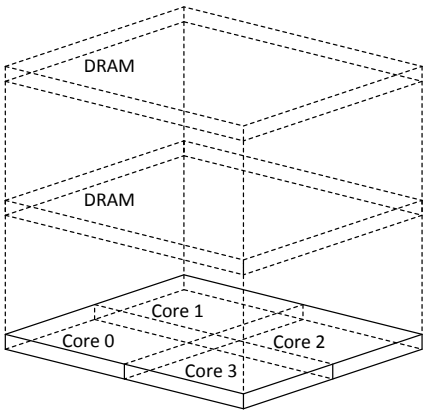


Fig. 3. SRAM-stacked 3D architecture. Multiple layers of DRAM can be stacked on top of the logic, which contains 4 processor cores.

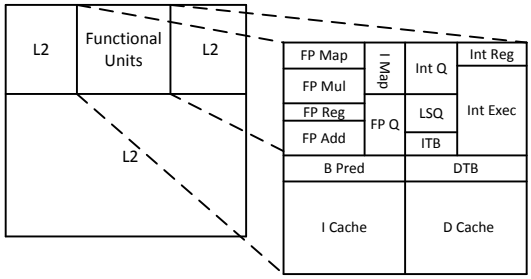


Fig. 4. The floorplan of Alpha 21264 processor

cache misses. However, the L1 cache misses can be serviced either by the coherence mechanism from the other private L1 cache, or from the shared LLC. In both cases, the cost is low. The TLB state is also lost due to thread migration. The TLB misses may be more costly without a shared TLB structure at the LLC level.

Researchers has also proposed "affinity scheduling" for parallel programs [16]. If we revisit this problem from the thermal perspective, we may decide against it because threads that are sharing resources tends to have similar thermal behavior. Scheduling them on the same processor core will cause heat to accumulate more quickly. Also because the shared LLC reduces the cost to re-create the contents of L1 cache, the benefit is getting smaller.

IV. Methodology

In order to evaluate the thermal behavior, we use the McPAT power model with the SMTSim [17] simulator to collect the power trace from the processor cores. We also add the DRAM power model derived from Micron data sheet[18] to the Gem5 [19] simulator. We combine the processor core power trace and the DRAM power trace to make the complete power trace for the 3D processor. Then we use the HotSpot [20] tool to calculate temperatures. We also add the migration algorithms to HotSpot to evaluate peak temperature and thermal variance reduction.

A. 3D multicore architecture

The 3D multicore architecture with the DRAM on top of the processor cores can provide higher memory bandwidth, shorter memory bus delay, and smaller energy consumption in the interconnection. To take advantage of these benefits, we have to face the thermal challenge. To investigate the thermal problem, we propose a simple 3D architecture, as is shown in figure 3. This architecture has been proposed in [21]. We could stack several layers of DRAM in this architecture. The DRAMs communicate with the processor with TSVs, which aren't shown in the graph. To simplify the problem, we model one layer of DRAM only. Note that additional layers of DRAM stacked on top of the processor increases the power density significantly which is not desirable. But our model is built to support multiple DRAM layers.

We use the Alpha 21264 floorplan for the processor core, which is shown in figure 4. We scale the original floorplan to 45nm with linear scaling. The floorplan is 6.4mm x 6.4mm. We constructed the 16-bank floorplan for the DRAM by evenly dividing the chip area into 16 squares. Hynix has demonstrated a 4Gb DDR3 SDRAM with the die area of $30.9mm^2$ using 23nm technology[22]. Therefore it is reasonable to choose 4Gb as the size of this stacked DRAM.

To make our discussion easier, we name the bottom layer

TABLE I
ARCHITECTURAL PARAMETERS OF SMTSIM SIMULATION

Item	Parameter
Processor configuration	4-core out of order,
Issue,Commit width	4
INT instruction queue	32 entries
FP instruction queue	32 entries
Reorder Buffer size	64 entries
INT registers	64
FP registers	64
Functional units	4 int/ldst 2 fp
L1 cache	32KB, 4-way, 2 cyc
L2 cache (private)	2MB, 16-way, 15 cyc
L3 cache (shared)	8MB, 8-way, 30 cyc
L3 miss penalty	250 cyc
Frequency	2GHz
Vdd	1.0V

TABLE II
SIMULATION WORKLOADS

ID	Benchmark Mix
1	soplex, swim, vortex, vpr
2	bzip2, cactusADM, facerec, galgel
3	omnetpp, perlbench, checkspam povray
4	lbm, leslie3d, libuantum, lucas
5	mcf, mesa, mgrid, milc
6	gcc , gromacs, h264ref, hmmer
7	applu, apsi, art, bwaves

"logic layer" and the top layer "DRAM layer".

B. Power Model

To calculate temperature, we need to first calculate the power consumed by each circuit unit on the floorplan. Power trace of processor cores are collected with SMTSim simulator integrated with McPAT power model. Table I shows the architectural parameters used in SMTSim simulation. Table II shows the 7 4-thread workloads we simulate. The workloads consists of benchmarks with both high and low memory bandwidth requirement. To collect the DRAM power, we used the timing memory modeled in Gem5 simulator. We used the default architecture parameters for Gem5 because we are only interested in the memory trace. We use the DRAM power model proposed by Lin et al. [23]. We assume that the static power is constant with respect to accesses and evenly distributed across all banks. The dynamic power is proportional to the read and write bandwidth. We used the access counts at each bank to estimate per bank power. We did our own calculation based on Micron data sheet [18] and derived the following formula for DRAM power at bank i:

$$P_i(mW) = 5.85 + 753 \times BW_r + 671 \times BW_w. \quad (1)$$

Where, BW_r and BW_w represents the read and write bandwidth in GB/s. Compared with numbers in [23], these constants are smaller, which should be attributed to the 5 years of advancement in DRAM technology. We generated 10 different memory power traces, and 70 combined power traces to study the interaction between the logic and DRAM layers.

The sampling period for both the processor core and the memory is set at 100μs.

TABLE III
HOTSPOT PARAMETERS

Item	Parameter
Die thickness	30μm
Ambient temperature	40 °C
Convection capacitance	40 J/K
Convection resistance	50 K/W
Heat sink side	70 mm
Heat spreader side	50 mm
Interlayer material thickness	0.05 mm
Interlayer material conductivity	5.0 W/(m-K)

C. Thermal Model

We use HotSpot to calculate the temperature from the power trace. HotSpot has been widely used in research publications and it has become a standard thermal model. Based on the processor core floorplan and the DRAM floorplan, we are able to model the temperature at each logical component in the logic layer and each bank in the DRAM layer. The tool provides us with both a transient temperature trace and a steady-state temperature for a given simulation.

The transient temperature is calculated based on an R-C thermal network and it is updated at each sampling point. Our dynamic algorithms make migration decisions based on the instantaneous temperature at the time, therefore it uses the transient temperature. However, since the temperature changes slowly with respect to power changes, the effect of power at the current moment won't be seen until a long time later. That is why the steady state temperature is calculated by averaging the power and using R network only. The steady-state temperature reflects the temperature in the infinitely far future by keeping the current average power level.

Table III lists the HotSpot parameters.

D. Migration Algorithm Implementation

Migration algorithms are integrated into the HotSpot simulator. In the beginning of HotSpot calculation cycle, power trace is read from the trace file and stored in an array. We migrate the power numbers inside the array before calculation starts. The temperature of the previous iteration is used to make migration decisions.

V. SIMULATION RESULTS

We have simulated 70 workloads with 5 different algorithms. Results show that thread migration is effective in reducing peak temperature and thermal variance in the 3D stacked architecture.

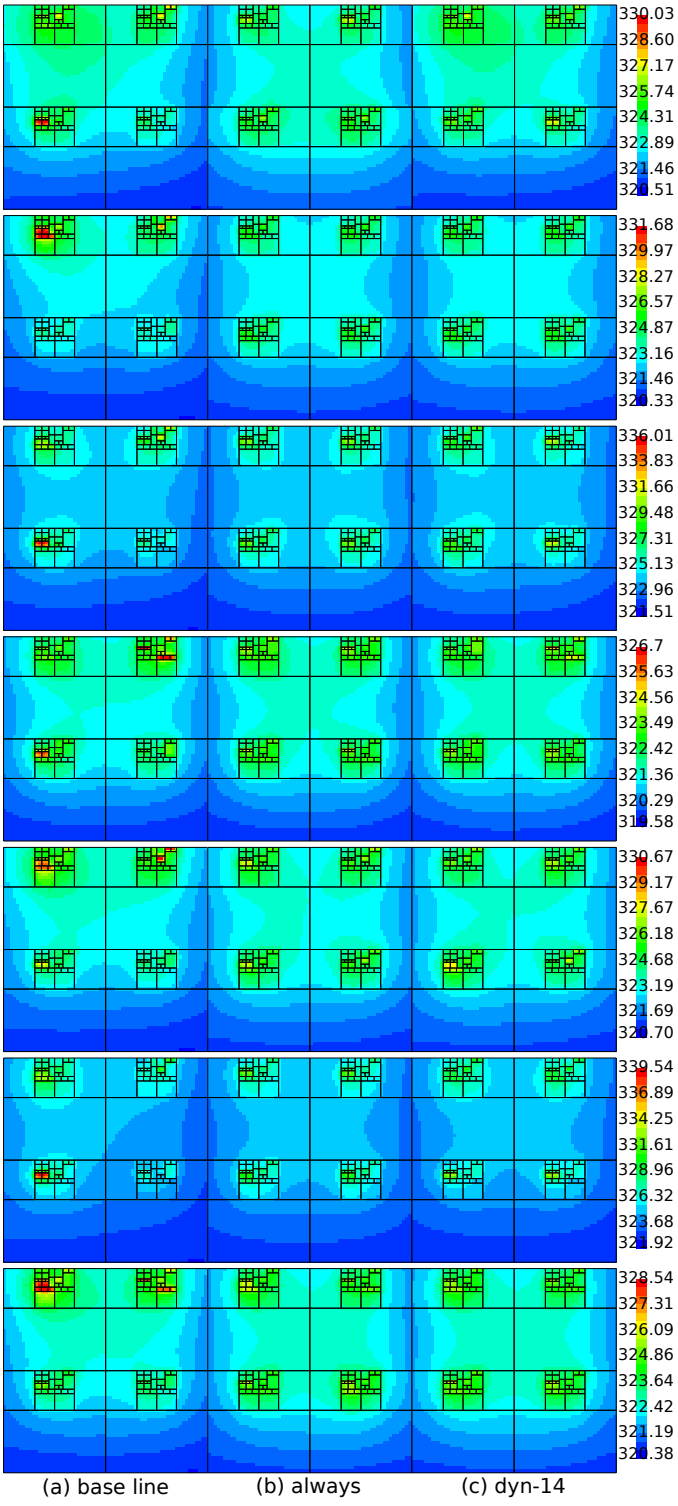


Fig. 5. Thermal map of steady state logic layer. 7 workloads x 3 algorithms.

A. Interaction between Layers

Table IV shows the average total power of the ten DRAM power traces and the seven processor core traces. We can see that the power consumption of the logic layer is more than 10 times larger than the DRAM. Not only is the DRAM power relatively small, as we have shown before in figure 2b, but its distribution is also uniform. Therefore, the peak temperature and thermal variance on the DRAM layer is mainly the result of the underlying logic layer rather than its own power dissipation. Figure 6 shows the steady state temperature of a 3D stacked design. The similar pattern in the DRAM and the logic layer confirms our earlier reasoning. Also, for the 10 different memory traces that is combined with the core power trace, the thermal map shows similar pattern with the logic layer. Therefore, we only presents the thermal map of our first 7 combined traces.

TABLE IV
POWER OF DRAM AND PROCESSOR (W)

ID	Logic	DRAM
1	68.61	7.41
2	68.98	5.54
3	79.66	5.68
4	56.75	7.38
5	72.02	5.78
6	86.29	4.52
7	64.46	8.72
8		7.21
9		5.78
10		5.96

B. DRAM Variance

Figure 7 shows the reduction in peak temperature in the DRAM layer by rotation and pairwise-14 with no dynamic optimization. The x-axis of the histogram represents difference in temperature (°C). We can see the peak temperature distribution gathering towards the lower side with both rotate and pair-wise algorithms.

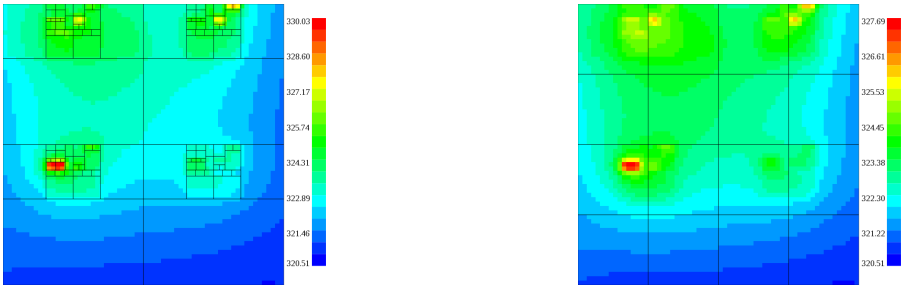
C. Logic Layer Peak Temperature

Figure 5 is the thermal map of the processor logic layer in the steady state. It shows all seven workloads with the three algorithms: baseline (no migration), rotation, and pairwise-14. We can clearly see that for all seven power traces, the logic layer has small hot spots that appear red. We can also observe that with thread migration algorithms, the red spots all disappear.

D. Logic layer Thermal Variance

Figure 8 summarizes the steady state thermal variance in the logic layer. Each one of the sevenworkloads are aligned on the x-axis. Y-axis represents the difference in the temperature (°C). The baseline variation ranges from 9 to 17 °C. The rotation algorithm does the best in reducing the variance. It achieves the most variance reduction except workload 2 with Pair-14 algorithm. The pair-14 algorithm performs slightly worse than rotation algorithm.

For dynamic algorithms, we need to choose the threshold for migration. According to figure 7, the rotation algorithm is able to reduce the variance to 2.0 degrees. Since dynamic algorithm



(a) Processor Cores (b) DRAM

Fig. 6. Thermal map of core and DRAM in 3D setting. Temperature is in Kelvin. Similar pattern confirms that the temperature of the DRAM layer is heavily influenced by the logic layer.

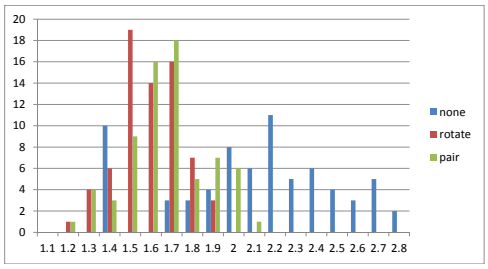


Fig. 7. Histogram of Variance in DRAM layer with baseline, rotation, and pair-14.

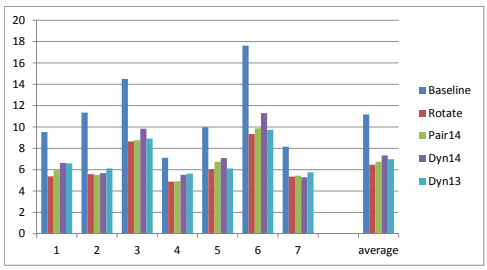


Fig. 8. Logic layer steady-state thermal variance

does not have to do better than the rotation algorithm, we choose 2.0 degrees as our threshold. The dynamic algorithm only migrates threads when the temperature difference between them is larger than 2.0 °C.

Our simulation show that using the threshold of 2.0 °C, Dynamic pair-14 algorithm on average reduces the number of migrations by 43%. Dynamic pair-13 algorithm generates 23% fewer migrations. Dynamic pair-13 is more aggressive than pair-14. This is consistent with figure 8 where the pair-14 has slightly higher variance than pair-13.

VI. Future Work

We have implemented a thermal-aware algorithm to migrate threads between processor cores to reduce steady state temperature and thermal variation in the DRAM and processor layer. Since the basic unit of movement is processor core that contains many functional units, of which some are hot and some are cold, In future we plan to study the migrate algorithm for memory activity to achieve a finer granularity of control.

The first approach is to put more memory banks than needed. Then there will always be some free banks, which should be cooler than others. So if somewhere becomes really hot, we could migrate the activity to the free cold bank. The second approach is to take advantage of cache replacement policy. The associativity of shared LLC is large enough so that we can pick victims that will be written back to the cold DRAM banks.

VII. Conclusion

Our study shows that in a 3D multi-core processor, the logic layer generates much more heat than the DRAM layer. The temperature of the DRAM layer is decided by the logic layer. Therefore we observed similar thermal variation in the DRAM layer, as the logic processor layer. To reduce peak temperature and thermal variance in DRAM layer, we propose various algorithms to migrate threads between processor cores. The overhead of our algorithm is small, with little/no impact on system performance. Yet we are able to reduce peak temperature by up to 8 degrees and reduce thermal variation in DRAM and processor layer significantly.

Acknowledgment

This work is supported by NSF grant CISE-SHF 1118047. The authors would also like to thank the anonymous reviewers for their useful feedback.

References

[1] A. L. Shimpi and B. Klug. (2011, Oct.) Apple iphone 4s: Thoroughly reviewed. [Online]. Available: http://www.anandtech.com/show/4971/apple-iphone-4s-review-att-verizon/5

[2] T. Liu, M. Li, and C. J. Xue, "Minimizing wcet for real-time embedded systems via static instruction cache locking," in *Proceedings of the 2009 15th IEEE Symposium on Real-Time and Embedded Technology and Applications*, 2009, pp. 35–44.

[3] I. Micron Technology. (2008, May) Technical note uprating semiconductors for high-temperature applications. [Online]. Available: http://download.micron.com/pdf/technotes/TN0018.pdf

[4] S. Liu, K. Pattabiraman, T. Moscibroda, and B. G. Zorn, "Flikker: saving dram refresh-power through critical data partitioning," in *Proceedings of the sixteenth international conference on Architectural support for programming languages and operating systems*, 2011, pp. 213–224.

[5] V. Bhalodia, "Scale dram sybsystem power analysis," Master's thesis, Massachusetts Institute of Technology, 2005.

[6] J. Lin, A. Oates, H. Tseng, Y. Liao, T. Chung, K. Huang, P. Tong, S. Yau, and Y. Wang, "Prediction and control of nbti – induced sram vccmin drift," in *Electron Devices Meeting, 2006. IEDM '06. International*, dec. 2006, pp. 1 –4.

[7] U. Kang, H.-J. Chung, S. Heo, D.-H. Park, H. Lee, J. H. Kim, S.-H. Ahn, S.-H. Cha, J. Ahn, D. Kwon, J.-W. Lee, H.-S. Joo, W.-S. Kim, D. H. Jang, N. S. Kim, J.-H. Choi, T.-G. Chung, J.-H. Yoo, J. S. Choi, C. Kim, and Y.-H. Jun, "8 gb 3-d ddr3 dram using through-silicon-via technology," *Solid-State Circuits, IEEE Journal of*, vol. 45, no. 1, pp. 111 –119, jan. 2010.

[8] M. S. Karthik Sankaranarayanan, Sivakumar Velusamy and K. Skadron, "A case for thermal-aware floorplanning at the microarchitectural level," *The Journal of Instruction-level Parallelism*, vol. 7, October 2005.

[9] K. Puttaswamy and G. Loh, "Thermal herding: Microarchitecture techniques for controlling hotspots in high-performance 3d-integrated processors," in *High Performance Computer Architecture, 2007. HPCA 2007. IEEE 13th International Symposium on*, feb. 2007, pp. 193 –204.

[10] S. Heo, K. Barr, and K. Asanovic, "Reducing power density through activity migration," in *Low Power Electronics and Design, 2003. ISLPED '03. Proceedings of the 2003 International Symposium on*, aug. 2003, pp. 217 – 222.

[11] S. Liu, J. Zhang, Q. Wu, and Q. Qiu, "Thermal-aware job allocation and scheduling for three dimensional chip multiprocessor," in *Quality Electronic Design (ISQED), 2010 11th International Symposium on*, march 2010, pp. 390 –398.

[12] J. S. Lee, K. Skadron, and S. W. Chung, "Predictive temperature-aware dvfs," *Computers, IEEE Transactions on*, vol. 59, no. 1, pp. 127 –133, jan. 2010.

[13] A. Coskun, J. Ayala, D. Atienza, T. Rosing, and Y. Leblebici, "Dynamic thermal management in 3d multicore architectures," in *Design, Automation Test in Europe Conference Exhibition, 2009. DATE '09.*, april 2009, pp. 1410 –1415.

[14] X. Zhou, Y. Xu, Y. Du, Y. Zhang, and J. Yang, "Thermal management for 3d processors via task scheduling," in *Proceedings of the 2008 37th International Conference on Parallel Processing*, 2008, pp. 115–122.

[15] K. Van Craeynest, A. Jaleel, L. Eeckhout, P. Narvaez, and J. Emer, "Scheduling heterogeneous multi-cores through performance impact estimation (pie)," in *Computer Architecture (ISCA), 2012 39th Annual International Symposium on*, june 2012, pp. 213 –224.

[16] J. Torrellas, A. Tucker, and A. Gupta, "Evaluating the performance of cache-affinity scheduling in shared-memory multiprocessors," *Journal of Parallel and Distributed Computing*, vol. 24, 1995.

[17] D. M. Tullsen, "Fellowship - simulation and modeling of a simultaneous multithreading processor," in *Int. CMG Conference*, 1996, pp. 819–828.

[18] I. Micron Technology. (2012, June) Micron mt41j256m16 ddr3 sdram datasheet. [Online]. Available: http://www.micron.com/~/media/Documents/Products/Data$%20$Sheet/DRAM/4Gb_DDR3_SDRAM.pdf

[19] 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 Comput. Archit. News*, vol. 39, no. 2, pp. 1–7, Aug. 2011.

[20] K. Skadron, M. R. Stan, K. Sankaranarayanan, W. Huang, S. Velusamy, and D. Tarjan, "Temperature-aware microarchitecture: Modeling and implementation," *ACM Trans. Archit. Code Optim.*, vol. 1, no. 1, pp. 94–125, Mar. 2004.

[21] D. H. Woo, N. H. Seong, D. Lewis, and H.-H. Lee, "An optimized 3d-stacked memory architecture by exploiting excessive, high-density tsv bandwidth," in *High Performance Computer Architecture (HPCA), 2010 IEEE 16th International Symposium on*, jan. 2010, pp. 1 –12.

[22] K.-N. Lim, W.-J. Jang, H.-S. Won, K.-Y. Lee, H. Kim, D.-W. Kim, M.-H. Cho, S.-L. Kim, J.-H. Kang, K.-W. Park, and B.-T. Jeong, "A 1.2v 23nm 6f2 4gb ddr3 sdram with local-bitline sense amplifier, hybrid lio sense amplifier and dummy-less array architecture," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2012 IEEE International*, feb. 2012, pp. 42 –44.

[23] J. Lin, H. Zheng, Z. Zhu, H. David, and Z. Zhang, "Thermal modeling and management of dram memory systems," in *Proceedings of the 34th annual international symposium on Computer architecture*, 2007, pp. 312–322.