Improving the Reliability of On-Chip Data Caches Under Process Variations

Wei Wu
Dept. of Computer Science
Univ. of Calif., Riverside, 92521
wwu@cs.ucr.edu

Jun Yang
ECE Department
Univ. of Pittsburgh, 15260
junyang@ece.pitt.edu

Sheldon X.-D. Tan
Dept. of Electrical engineering
Univ. of Calif., Riverside, 92521
stan@ee.ucr.edu

Shih-Lien Lu
Intel Corporation
Hillsboro, OR 97124
shih-lien.l.lu@intel.com

Abstract

On-chip caches take a large portion of the chip area. They are much more vulnerable to parameter variation than smaller units. As leakage current becomes a significant component of the total power consumption, the leakage current variations induced thermal and reliability problem to the on-chip caches become an important design concern.

This paper studies the impact of process variations, particular the leakage variations, on the temperature and reliability of on-chip caches. Our statistical simulation shows that, under process variation, 85% of the caches see shortened lifetime, with average lifetime being 81.6% of the ideal cache. At runtime, unevenly distributed dynamic power and the corresponding thermal variation would further deteriorate the situation. To mitigate this problem, we propose a dynamic cache subarray permutation scheme that can alleviate the thermal stress on a high-leakage area to improve the reliability of the caches. Experiments on 17 Spec2k benchmarks show that our scheme can extend the cache lifetime by up to 20.3%, and reduce the peak temperature by 7 degrees on average and more on data-intensive applications.

1 Introduction

As technology scaling approaches to the nanometer scale, high performance microprocessor designers are facing two major challenges. The first is the increasing variations due to manufacture processes and runtime variability. One important problem associated with variations is the potential yield loss and reduced lifetime and reliability. The second challenge is the increasing power density partially due to large leakage power [12]. The variation in fabrication process directly affects device parameters such as transistor channel length and gate oxide thickness [7]. These will change the circuit behavior which ultimately affects its power and performance, and increases the probability of failures [18]. In addition to this one-time variation due to fabrication, there are runtime variations, such as distributed runtime temperatures, device ware out, and voltage drop variation in power supply networks. As a result, design processes must consider the variations to make robust and reliable chips.

Modern on-chip caches in today's microprocessor occupy a significant portion of the total chip area. As a result, it is particularly vulnerable and exposed to significant amounts of variations [17]. Recent industry studies show that the difference in total leakage power from die to die on the same wafer can be as large as 20X [7]. Likewise, the leakage current within a die also varies greatly even for schematically equivalent devices like memory cells. Highly leaked devices consume more power which increases temperature faster than normal devices due to the exponential relation between leakage power and temperature.

High temperature leads to shorter lifetime, as most hardware failure mechanics are exponentially related to temperature [4]. Recent study shows, multi-ported data caches in superscalar microprocessors are very often exposed to high-temperature[15]. To extend the lifetime of a cache, the runtime temperature should be reduced, especially the peak temperature. Conventionally, redundant rows/columns of cache cells were used to replace the faulty rows/columns [11]. However, they are not run-time techniques, and are used during testing [24] before shipping the chips. It was suggested that redundancy could be exploited to increase processor lifetime, but at a cost of runtime performance [2]. In this paper, we assume a whole cache becomes faulty if one cache block is faulty.

Recently, there have been a number of techniques proposed to reduce the temperatures and the leakage power of on-chip caches. The main idea is to distribute the power evenly in a cache to reduce its power density. For example, cache blocks are permuted such that the spatial locality does not result in high access rate in one cache subarray [16, 13]. Another solution is to create interweaved hot and cool cache blocks by disabling idle blocks [16]. However, none of the proposed techniques considers the impact of process variation. Since different cache blocks have intrinsic differences in their leakage current, and at runtime they may be stressed differently by different workloads, they should no longer be treated as the same. Instead, an adaptive strategy to runtime temperature change should be adopted, taking into account the fabrication variation factor.

In this paper, we examine the cache thermal and reliability issues considering process variation, especially the leakage induced variations. The main contributions are as follows:

- We develop an architecture level method for evaluating power, thermal and reliability of caches in the presence of process variation. Particularly, the leakage power model, which consists of sub-threshold and gate leakage, includes parameters for both device gate length and temperature. This is in contrast to the gate-length-only model in [17].
- We perform statistical Monte Carlo simulation on 10,000 samples of the cache, to study the effect of process variation and show that 85% of the caches have shortened lifetime with average lifetime being 81.6% of an ideal cache.
- We develop a *dynamic* (runtime) cache block permutation scheme, in contrast to a *static* scheme [16] where permutation is done in one time.
- We extend the average MTTF of a cache by up to 20.3%, reduce its total leakage power and peak temperature by 7 degrees on average.

The remainder of the paper is organized as follows. Section 2 introduces the models. Section 3 analyzes the performance of cache under parameter variations. Section 4 describes our cache design. Section 5 and 6 present our experiment framework and results. Finally, Section 7 concludes this paper.

2 Process variation and cache reliability modeling

To evaluate the cache reliability under process variation, we developed an architectural level methodology with a number of models. The detailed approach, depicted in Fig.1, is comprised of three main steps: power, temperature and reliability. The power model is comprised of two components, dynamic power and leakage power. Leakage power depends on both temperature and gate length. We adopted a widely used hierarchical spatial correlated model to generate gate length variation over the whole cache. Then based on power consumption, temperature is simulated over reasonable time intervals by using a fast simulation method. At last, having a temperature trace of entire simulation, cache reliability of each workload can be calculated as an average over time and space.

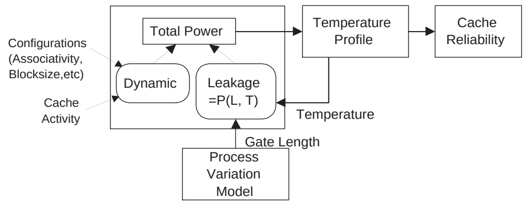

Fig. 1: The complete system of models used in this work.

In the following, we give detailed descriptions for each model.

2.1 Power model

A cache circuit model is built in PTM (predictive technology model) 45nm technology [19], and simulated in HSpice. It gives dynamic and leakage power of an ideal cache with uniform SRAM cells and gate length. We also generate a leakage power model based on one standard 6T-cell, with two parameters: temperature and gate length.

Generally, power dissipation can be divided into two parts: dynamic power and leakage power.

Dynamic power, also called switching power, is consumed whenever cache is accessed. As components change their states, corresponding load capacitances will be charged or discharged. So $P_{dynamic}$ is proportional to cache access frequency, which can be logged during architectural simulation. Notice that dynamic power is temperature-independent as capacitances are constant under various temperatures. And we also neglect the process variation in load capacitances. Therefore the dynamic per access power can be treated as a constant.

Leakage Power, as technology scales down, becomes more and more dominant in total power dissipation.

Among various components in cache, cell array consumes the bulk of power and occupies most of the cache area, so we focus our leakage study on the cache cells. The total leakage is proportional to the number of cells, as estimated in Eq.1:

$$\begin{aligned} P_{total} &= N_{cell} \times V_{dd} \times I_{leak\_6Tcell} \quad (1) \\ &= N_{cell} \times V_{dd} \times (I_{gate\_leakage} + I_{subthreshold\_leakage}) \end{aligned}$$

Fig.2 shows a 6-T SRAM cell with one read-write port. The bit value is stored and reinforced by the two cross-coupled inverters. In Fig. 2, we show all leakage paths for this cell when storing a '1'. The structure is symmetric, so the leakage for storing '0' is the same.

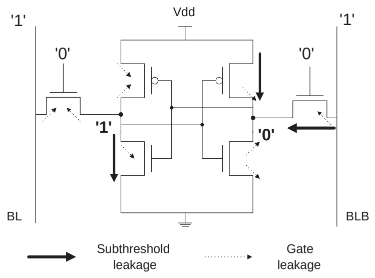

Fig. 2: Leakage paths for a 6-T SRAM cell.

The leakage power model for each cell is obtained as follows. We did multiple simulations, with temperature ranging from 30°C to 120°C, and step size is 10°C. Under each temperature, the gate length varies from 90% to 110% of the normal value. The results are shown in Fig. 3. All values are normalized to the subthreshold leakage power of a regular sized cell at 40°C. In this figure, there are two groups of lines, the lower group is for gate leakage, which is relatively constant under different configurations. The reason is that the oxide thickness is the only parameter that significantly impact the gate leakage [8]. The upper group is the subthreshold leakage. This group clearly shows the exponential growth of leakage with temperature. The 11 lines correspond to 11 different gate lengths. The smaller the gate length, the higher the leakage power. According to [10], the subthreshold leakage can be expressed as :

$$I_{leakage} = A(l) \times T^2 \times e^{-B(l)/T} \quad (2)$$

A and B used to be constants as modeled in [10], However, as the gate length varies, the leakage current curves start to diverge. We can use a 2nd order polynomial function of length to represent this behavior:

$$A(l) = p_2 \times l^2 + p_1 \times l + p_0 \quad (3)$$

and $B(l)$ is modeled similarly. Finally, we derived all coefficients of the l terms by fitting the curves with the calculated values. The resulting model achieves an error rate of less than 1%, as shown by the symbol $\star$ in Fig. 3.

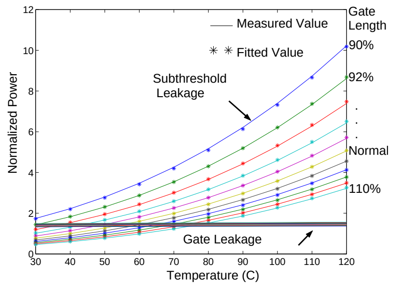

Fig. 3: Leakage power curve fitting.

2.2 Process variation model

Among many circuit parameters, the gate length plays a dominant role in subthreshold leakage [20]. In our methodology, the process variation model generates gate length variation for all devices over the whole cache. The length information will then become the input of the leakage model to calculate total leakage power.

According to recent studies, intra-die variation is more dominant than inter-die variation as technology size scales down [3]. So in this paper, we only considered the intra-die variations. Electrical measurements of a full wafer showed that intra-die gate length variation has strong spatial correlations [9]. This means that devices that are physically close to each other are more likely to be similar than those that are far apart. So we use the hierarchical correlation modeling [14] widely used in CAD area to capture the length variation among the devices.

Our model could be enhanced by considering more kinds of process variations, such as the Vt variation due to random dopant fluctuations. However, current model is sufficient to show the non-uniform characterization lies in on-chip cache. A more detailed variation model can only reinforce such non-uniformity, and therefore providing more opportunities for our proposed technique.

2.3 Thermal model

The thermal model used in this paper includes several parts, the die which generate power and therefore the heat, the interface material; which connects to heat spreader, and heat sink. The thermal circuit model is generated by HotSpot 3.0 [21], as it provides grid-based model which is useful for our cache block.

We solved the thermal circuit model using a rapid temperature calculation algorithm, called Thermal Moment Matching(TMM) [6]. The TMM calculated the transfer function from power to temperatures in frequency domain, which is much faster than the integral-based method used by HotSpot. As we need to do the thermal simulation and a large number of Monte Carlo simulation, a fast simulator is very important. As leakage power is a variable to temperature, for each time interval, we call the temperature function multiple times, until the leakage becomes constant.

2.4 Reliability model

Processor errors can be generally classified into two categories, soft errors and hard errors. Soft errors, a.k.a. transient faults, occur intermittently. Hard errors, on the other hand, are permanent and physically damage the chips once the occur. In this paper, we only concern the reliability problems due to hard errors. Srinivasan et al. proposed an architectural reliability model, called RAMP [4]. The Mean-Time-To-Failure (MTTF) is used as a metric for evaluation. RAMP models four different hard-failure mechanisms. Three of them have exponential dependencies on temperature.

As we divide the cache into a grid of cells to model the process variation distributions, the temperature of each grid cell is different throughout the simulation. To calculate the total MTTF for a whole cache, we use the same assumption as in RAMP: (1)The cache is a serial failure system, which means the first instance of any failure causes the entire circuit to fail, (2) each individual failure mechanism has a constant failure rate, and follows an exponential lifetime distribution. Based on the above assumptions, total MTTF of the cache is the inverse of the sum of failure rate of all individual cache cells of all four failure mechanisms.

$$MTTF_{cache} = \frac{1}{\lambda_{cache}} = \frac{1}{\sum_{i=1}^{n}\sum_{j=1}^{4}\lambda_{ij}} \quad (4)$$

where λ_{ij} represents the failure rate of the i^{th} cell due to the j^{th} failure mechanism. When determining the MTTF of the cache over a complete workload, the failure rate is averaged over time on a number of time intervals and the temperature within one interval is considered constant. As in [5], we assume the four failure mechanisms contribute equally to MTTF_{cache} at 80 degrees.

3 Analyses of a cache under process variation

With process variations, the leakage power in a cache structure becomes increasingly non-uniform. Localized high power density causes the temperature to increase well above average, and warm up the neighboring circuits as well. High temperature exponentially increases the circuit failure rate, and in turn shorten the circuit life. Therefore, caches with parameter variation are facing a more urgent reliability problem than circuit designers thought. In this section, we study both the local and global effects of process variation on a cache. Results are analyzed using three metrics: leakage, temperature and reliability.

When simulating the temperature, we divide the cache into a grid of 16 by 16, and treat the temperatures of all gates inside one grid identical. Then we assume an average cache access rate of 10%, which is an approximate average of 20 Spec2K benchmarks simulated in SimpleScalar. Dynamic power is assumed to be evenly distributed over the whole cache, and there are no lateral thermal resistances between other function blocks and cache. Steady temperature of each cache grid cell is collected.

In Fig. 4 and 5, we show the steady temperature profiles for a cache without and with process variation respectively. Fig. 4 has a rather balanced temperature distribution. The center is warm, and the peripherals are a little bit cooler. The maximum difference is within 0.5 degree. Fig.5 is based on a cache with length variations. Its bottom right corner is very hot, but the bottom left corner is almost 12 degrees cooler. As a result, the profile shows a totally different distribution. Although two caches have similar overall average temperature, their peak temperature varies greatly.

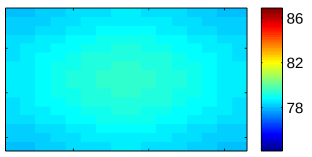

Fig. 4: Temp. profile for an ideal cache.

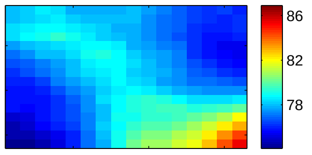

Fig. 5: A sample of cache with gate length variation.

We use the Monte Carlo method to evaluate a total of 10,000 cache samples. Such a sample size represents $\sim$ 1% error for estimating the standard variation or variance. Fig. 6 shows the leakage power of each grid at their steady temperature. Values are normalized to that of a normal cache that is shown in the left bar. Each cache grid cell has same area, so a larger leakage means a larger power density which leads to higher temperature. Fig.7 shows the steady temperature statistic information.

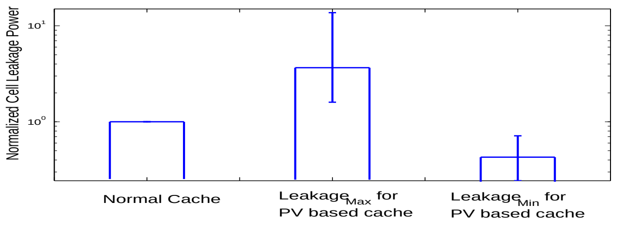

Fig. 6: Normalized leakage power for all cache grid cells. ***$Leakage_{max}$*** **$\in$(0.71, 13.72) with an average of 3.66.** ***$Leakage_{min}$*** **$\in$(0.27, 0.73) with an average of 0.42.**

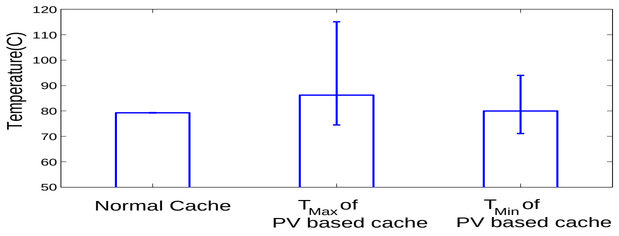

Fig. 7: Steady temperature for cache grid cells. Average temperature of a normal cache is 79.3°C. ***T_{max}*** **$\in$(74°C, 115°C) with an average of 86°C.** ***T_{min}*** **$\in$(71°C, 94°C) with an average of 79.9°C, which is 0.6 degree higher than regular value.**

With above information, we show the reliability of the cache in terms of MTTF. Fig 8 shows the MTTF distribution normalized to an ideal cache. Here, the y-axis means the number of occurrence among the 10,000 samples, and the x-axis means the MTTF values normalized to an ideal cache. The smaller the MTTF value means the shorter expected life time. Totally, 85% of the 10,000 simulated caches have shortened lifetime. The mean value is 0.816, which indicates that the average life time is shortened by 19%. The worst case could have a lifetime of only 30% of a cache without process variation.

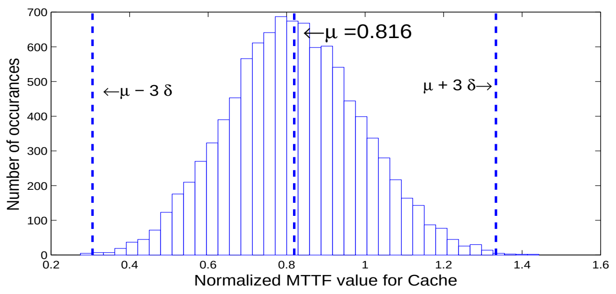

Fig. 8: Distribution of normalized cache MTTF.

These statistic data reveal that, the cache under the process variation not only has the thermal emergencies due to local high power density, but also faces serious reliability problems. In addition, the situation could be even worse under real workloads since some of them may have large cache access rates, and hit in a high leakage region frequently. In order to alleviate such thermal and reliability stress, we propose a method to map the logic cache subarrays to different physical positions in the cache, such that dynamic cache accesses do not always occur in high-leakage area. Our scheme has little overhead and can be performed during runtime.

4 Thermal aware dynamical permutation cache design

We have shown in the previous section that the process variation induced leakage variation alone can generate unevenly distributed temperature across the entire cache. This situation can be even worsened by the unbalanced cache subarray accesses that contribute to the dynamic power. In the worst scenario, a highly leaked area could be highly accessed at runtime as well. Motivated by this observation, we propose a dynamic runtime subarray permutation scheme with an objective to balance the temperature distribution in a cache.

There have been a number of techniques proposed to reduce the temperatures and the leakage power of on-chip caches. The main idea is to distribute the power evenly in a cache to reduce its power density. For example, cache blocks are permuted such that the spatial locality does not result in high access rate in one cache way [16, 13]. Another solution is to create interweaved hot and cool cache blocks by disabling idle blocks [16]. However, none of techniques considered the impact of process variation. Since different cache blocks have intrinsic differences in their leakage current, and at runtime they may be stressed differently by different workloads, they should no longer be treated homogeneously. Static permutation schemes as proposed in [13] and [16] cannot solve this problem. Instead, an adaptive reaction to runtime temperature change should be adopted, taking into account the fabrication variation factor.

The main idea of our scheme is to move the high access rate from hot subarrays to cool subarrays. As the cool regions warm up, the hot regions can have a chance to cool down. Hence, at runtime, we periodically check whether certain conditions are satisfied. If so, we permute the predecoded address lines, such that the logic subarray positions are mapped to different physical positions. Data stored in the original place will be lost, but the time interval is long enough to ignore the cache warm-up time. The detailed scheme has some adjustments to reduce false permutations. We assume, in future processors, large amount of thermal sensors will be embedded on-die, so we can get run-time temperature information to control the permutation scheme. In the following, we first explain the hardware design, and then explain the control algorithm.

4.1 Dynamic Subarray Permutation Circuit

We discuss how we modify the predecoded address signals. First, we show a simplified cache data array structure in Fig.9. The cache here has four ways, each divided horizontally into 8 subarrays.

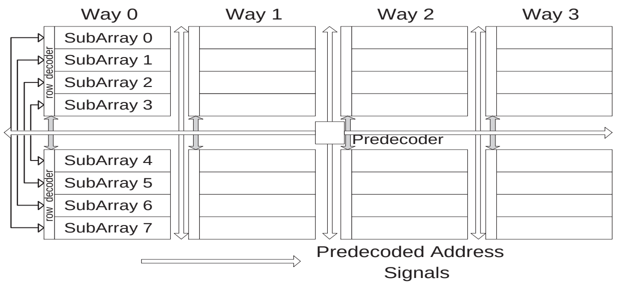

Fig. 9: General structure of the cache data subarrays

The small block in the center is a predecoder. In order to overcome the pitch-matching problem and reduce wire capacitance, the address decoder is usually divided into two stages: predecoder and row decoder. The predecoder identifies which subarray should be accessed for a given address; and then the row decoder selects an appropriate cache line inside that subarray. Thus, at any time, at most one of the eight arrows in front of way 0 will become '1'.

The subarrays inside each way are always placed in order. As shown in Fig.9, the one at the top is always subarray0, and the one at the bottom is subarray7. This is determined by the wiring of the decoder. Since there are eight subarrays, three address bits are needed to index them. The predecoder can thus be implemented using a 3-to-8 decoder with sorted outputs from 0 to 7. For example, when the address bits are "000", the decoder always activates the top most subarray0.

Our design is to insert 3 sets of crossbars into the predecoded address lines, as depicted in Fig. 10. Each set is comprised of 4 mini-crossbars. The function of crossbar is depicted at the bottom of the figure. When the SEL signal is set to '0', meaning no crossover, the output is the same as the input; when set to '1', the two inputs are swapped in the outputs. The leftmost crossbar swaps the address lines with

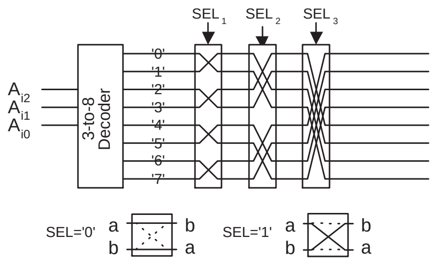

Fig. 10: Crossbar structure for subarray permutation

distance of 1, the middle one with distance of 2, and rightmost one with distance of 4. For example, when SEL_{321} is set to "101", it will not only swap the odd and even blocks, but also move the lower 4 subarrays to the top.

There are totally eight (2^3) different configurations. With three SEL signals being set or clear, we can move one subarray to any desired position. Having different SEL configurations for each way, two subarrays on same logical row in neighboring cache ways can be put far apart.

At runtime, when we pick out two physical subarrays to swap, we do not need to know their logical row numbers. The new SEL signal is determined by:

$$SEL_{new} = SEL_{old} \oplus (Position_a \oplus Position_b) \tag{5}$$

The second XOR counts the bit difference between two blocks' position, and determines whether each crossbar has to be set or not. The first XOR applies the difference on top of current configuration.

4.2 Control Algorithm

We need a simple and lightweight control scheme to determine when and how the permutation should be performed. We assume there are enough thermal sensors to measure the temperatures, and enough performance counters to log the access rate of each subarray. Then for every 1ms, we invoke the control algorithm once to check is there any permutation required. The time interval is set to 1ms as OS performs some housekeeping task on this time interval. As we will explain, the algorithm entails only some adders, comparators, and logic gates. We can implement it all in hardware. Thus the software overhead will be omitted in the evaluation.

During each time interval:

- First, we select the hottest and coolest subarray inside each cache way.
- Second, if the temperature difference is more than 5°C, we perform a permutation.
- Calculate the new permutation setting from equation (5).

The 5°C is an empirical threshold to trigger a permutation. The smaller the threshold, the more sensitive the algorithm is to temperature variations, and more frequent the permutation.

However, we have additional constraints to eliminate false permutation (or simply swapping). False swapping happens when high-temperature is caused by high leakage instead of dynamic power, and swapping cannot change the distribution of leakage power. Such swappings are redundant, and degrades the performance. We add the following two constraints:

- If the hot subarray's temperature has been reduced since last interval, we will keep the old configuration.
- If the access rate of hot subarray is less than that of the cool block, or less than 5% in general, we choose the second hottest subarray to swap instead.

The first permutation prevents overly frequent permutations. When the current configuration does take effect in reducing the peak temperature, we should simply keep it as is. The second one tries to avoid doing false permutation.

4.3 Overhead Discussion

The hardware overhead lies in two parts. This first part is the permutation circuit which consists of three stages of crossbars for each cache way. Each crossbar has 16 pass-transistors, which are simply pairs of NMOS and PMOS gates. The area overhead of these pass-transistors can be hidden in the existing space slacks within the eight predecoded address signals. This is because those signal wires are fairly long, so they have intermediate large inverters to drive them, and the wires are thus separated apart to accommodate those drivers. These spaces along the path can be exploited to place the pass-transistors. Therefore, the added transistors on the path will not be a significant portion to the total cache area. Timing wise, the delay on the pass transistors can be shortened by increasing either the transistor sizes or the driver sizes.

The second part is the circuit for the control algorithm. According to the scheme, we need registers to hold the peak temperatures of the previous interval. Comparators are also needed to find out the hottest and coolest subarray; XOR gates are needed to calculate the new 'select' signals. These circuitry can be put on the side of the cache, and only be invoked on millisecond time interval. Hence, their timing overhead is insignificant to the overall program's performance.

As the control algorithm can be implemented in hardware, the performance degradation is mainly dominated by flushing the cache after each permutation. The granularity of permutation is on the millisecond level. We collected the number of permutation for 17 Spec2K benchmarks. The detailed experiment setting will be explained in the next section. The overall average permutation frequency is 1.06/ms, which means on average, every millisecond, there is one cache way being flushed out. From our experimental experience, L1 cache warm up time is typically on the order of tens of thousands of cycles. Therefore, flushing a portion of the L1 data cache once every millisecond on a gigahertz processor hardly has any effect on the performance of the program execution.

5 Experiment setting

To evaluate our proposed techniques, we use SPEC2K benchmarks, simulated in SimpleScalar [1]. The processor configuration parameters are listed in table 1.

Issue width	8 instructions
Number of RUU	128
Number of LSQ	32
Branch Prediction	Bi-modal with 2048 entries
Integer ALUs (Mult/Div)	8 (2)
FP ALUs (Mul/Div)	8 (2)
L1 data cache	64KB, 4way, 32B blocks, 2 cycles
L1 instruction cache	64KB, 4way, 32B blocks, 2 cycles
L2 cache	512KB, 8way, 128B blocks, 16 cycles
Memory access time	First access: 250 cycs Subsequently: 6 cycs
Memory bus width	8 bytes

Table 1: Processor configurations

The number of reads and writes for each subarray were recorded during the simulation. For all the benchmarks, the simulator was run for 500 millions instructions after fast-forwarding a specific number of instructions determined by SimPoint [22].

To calculate the temperatures, we use a time epoch of 1ms, far less than a processor's thermal time constant which is around 10 milliseconds [15]. Also, 1ms is the typical interrupt period in an operating system. We assume the processor clock frequency is 4GHz.

As for the process variation cache model, we picked one from the 10,000 samples that has the leakage and MTTF around the mean values of all samples. Thus, it can represent a typical process variation scenario. Fig. 11 shows some details of this model. This cache has a similar configuration as the data cache in Alpha 21364. It has 64KB, 4-way and 32-Byte cache lines. According to CACTI 3.2 [23], the optimal parameters for this cache are Ndwl=4, Ndbl=2, and Nspd=1. With this configuration, each cache way occupies on a column in the figure, and each way is broken into upper and lower parts. And we further divide each part into four subarrays. The darker area represents lower leakage power, and brighter area means the opposite. Overall, the upper half is lighter than the lower half. And the top right is even more leaky.

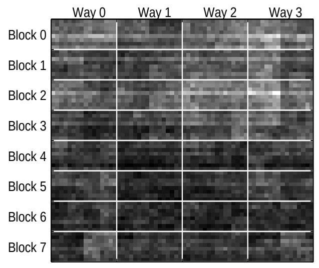

Fig. 11: Experiment model

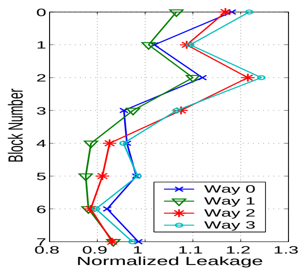

Fig. 12: Subarray leakage

To further quantify the leakage, in Fig. 12 we show the total leakage power for each subarrays at 60°C. The values are normalized to leakage power of a subarray having a regular gate size. Due to the spatial correlation, the curves look quite similar between different ways. Subarray 0 and 3 are most leaky, and subarray 6 has the lowest leakage.

6 Experimental results

In this section, we evaluate the effectiveness of our proposed dynamic permutation scheme, and compare it to the baseline in-order subarray placement (labeled 'Original'), and the statically permutation placement (labeled

'Static') [16]. The initial temperatures for each benchmark are set to the steady temperatures of the conventional cache.

We did not apply common low-power techniques such as the selective cache ways [16], cache-decay [13] etc. Such techniques not only save power, but also provide a lot of idle and cool blocks, so that the remaining working blocks can distribute their heat. In our experiments, we show that in the worst case where no blocks are shut down for power savings, we can achieve a large peak temperature reduction, and extend the cache's lifetime.

Fig.13 shows sample temperature variations for Bzip2 benchmark. We show the temperatures of 8 subarrays in way2 since other 3 ways are similar. The solid lines are for the baseline cache. Among them, two lines are much higher than the others because of the high access rates of subarray 5 and 6. The dotted lines show the result of our dynamic scheme. As we can see, when detecting a high temperature block, our scheme swaps it with a cool block, reducing the temperature in the hot block effectively. As a results, the temperatures in all subarrays are fairly close to each other and around their average.

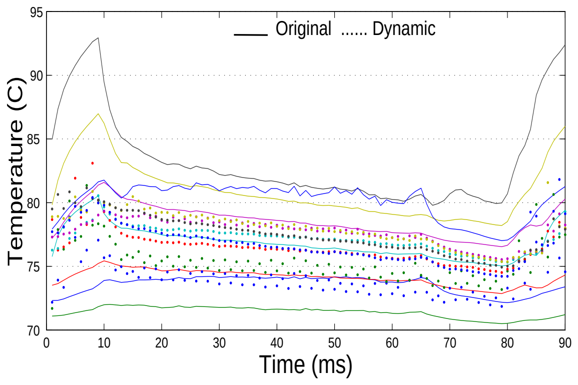

Fig. 13: A sample result of Bzip2– temperatures of 8 subarrays in way 2.

In the following figures, we show the result of three different schemes: Original, Static, and Dynamic which is our proposed permutation scheme. In the Static scheme [16], the subarrays are permutated in fabrication and before the shipment of the chip. Therefore, the configuration is hardwired in circuit. The placement follows the rule that logically neighboring subarrays are separated as much as possible, and same subarrays of different ways are also separate.

In Fig. 14, we show the peak temperatures of the three schemes. As expected, our Dynamic scheme achieves the largest reduction in peak temperature: on average a 7.05°C drop from the baseline design. The Static scheme also reduces the peak temperature by 2.2°C on average. We also find that for Lucas , the peak temperature even increases a little. This is because it has a very low cache access rate, so most of the power consumption (more than 80%) are due to leakage. If one block is hot, it is caused mostly by the leakage not dynamic power, although a cool block may have more accesses and dynamic power. Therefore, when block placement is rearranged, it often made the wrong decision and increased the dynamic power of an already relatively hot block.

For static scheme, many benchmarks e.g. apsi, gcc show higher peak temperature than baseline design. This is because static scheme has no selection for workload to subarrays. Its improvement on reducing temperature is not stable, and depends on workload.

We remark that our dynamic permutation scheme work particularly well with data intensive benchmarks. Among the 17 benchmarks, we picked those with average cache access rate above 5%, the average peak temperature reduction can achieve 8.5 degrees.

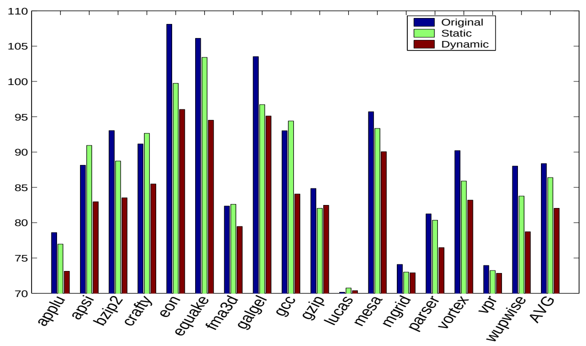

Fig. 14: Peak temperature comparison

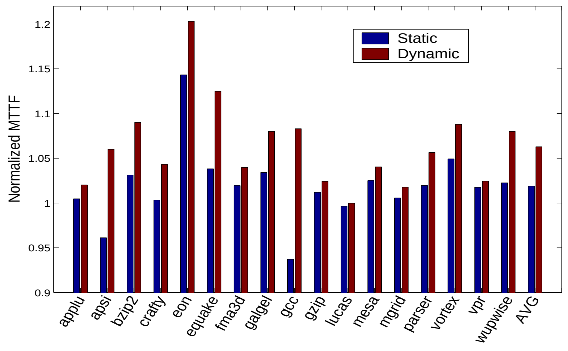

Fig. 15: MTTF comparison

In Fig. 15 we show the MTTFs of the cache when running different benchmarks. For each benchmark, the values of static and dynamic schemes are normalized to that of the original cache in baseline (no permutation). When considering the reliability, the subarray with peak temperature usually plays a more dominant role. This is because if one subarray is constantly much hotter than others, it will have a higher hard failure rate, which will lead to sooner failure of the cache (when one subarray fails, the cache activities cannot be trusted so the entire cache is viewed as failed). Thus, those benchmarks with large peak temperature reductions show a better reliability improvement. Examples are Eon, Equake, Gcc etc. Among all test programs, Eon achieves the largest lifetime extension: 20.3% longer than the baseline cache. The average of all benchmarks is 6.3%.

Another property observed from Fig. 15 is that for all benchmarks, dynamic permutation scheme is better than the baseline design (higher than 1) and also static scheme. Even for Gzip, where dynamic scheme's peak temperature is slightly higher than static one. The reason is that the dynamic scheme aims to achieve a globally balanced temperature profile. Throughout the entire execution of a program, the temperatures of different subarrays are maintained within certain bound. So overall, it can always achieve a longer lifetime.

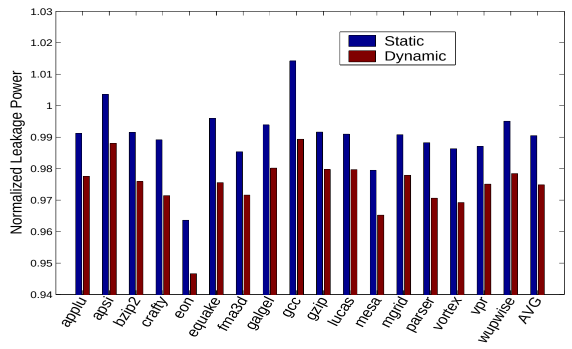

Fig. 16: Total leakage power, normalized to baseline cache

Finally, we show the leakage power reduction among three cache schemes. On average, the static scheme reduces the total leakage by around 1%, and the dynamic scheme reduces 2.6%. The leakage power reduction is not very prominent. There are several reasons for this. First, as we reduced the peak temperature, workload is shifted to cooler subarrays and therefore, the leakage on cooler subarrays are increased. Second, the leakage power variation within a narrow temperature range is close to linear, although in a much larger temperature range, it will become exponential with respect to temperature. To further reduce the total cache leakage, other low-power techniques such as selective cache way and cache decay should be applied.

7 Conclusion

In this work, we have studied the effects of the process variation on reliability and thermal issues of on-chip caches. To the best of our knowledge, our work is among the first to examine such effects of process variations. Our statistical simulations showed that parameter variations could significantly reduce the cache lifetime to 81.6% of its original lifetime on average. The cache temperature profiles show large gradients across the whole cache. The maximum leakage power density ranges from 0.71 to 13.72 times of the baseline value. To solve such a problem, we have proposed a dynamic cache subarray permutation scheme using crossbars in the address predecoder. Through dynamically changing the cache block placement, we can alleviate the thermal stress on high-leakage area, and reduce the peak temperatures inside a cache. We tested our scheme on 17 Spec2k benchmarks, the cache lifetime is extended by up to 20.3%, and the peak temperature is reduced by 7 degrees on average.

References

[1] Simplescalar tool. http://www.simplescalar.com.

[2] A. Agarwal, B. C. Paul, and K. Roy. A novel fault tolerant cache to improve yield in nanometer technologies. In *IOLTS*, pages 149–154, 2004.

[3] S. G. Duvall. Statistical circuit modeling and optimization. In *5th Intl. Workshop Statistical Metrology*, pages 56–63, June 2000.

[4] J. S. et al. The case for microarchitectural awareness of lifetime reliability. In *Proc. of the 31st Annual Intl. Symp. on Comp.Arch. (ISCA)*, 2004.

[5] J. S. et al. The impact of technology scaling on lifetime reliability. In *Dependable Systems and Networks*, 2004.

[6] P. L. et al. Fast thermal simulation for architecture level dynamic thermal management. In *(ICCAD)*, pages 639–644, Nov 2005.

[7] S. B. et al. Parameter variations and impact on circuits and microarchitecture. In *Proc. Design Automation Conf. (DAC 03)*, pages 338–342. IEEE Press, 2003.

[8] Y.-F. T. et al. Chippower: an architecture-level leakage simulator. In *SOC Conference, Proceedings. IEEE International*, pages 395– 398, 2004.

[9] P. Friedberg. Modeling within-die spatial correlation effects for process design co-optimization. In *Proceedings of the 6th International Symposium on Quality of Electronic Design*, pages 516–521, 2005.

[10] L. He, W. Liao, and M. R. Stan. System level leakage reduction considering the interdependence of temperature and leakage. In *DAC*, pages 12–17, 2004.

[11] M. Horiguchi. Redundancy techniques for high-density drams. In *Second Annual IEEE International Conference on Innovative Systems in Silicon*, pages 22 –29, Oct 1997.

[12] Semiconductor industry association. international technology roadmap for semiconductors, 2003. http://public.itrs.net/Files/2003ITRS/Home.htm.

[13] J. K. John, J. S. Hu, and S. G. Ziavras. Optimizing the thermal behavior of subarrayed data caches. In *International Conference on Computer-Aided Design*, 2005.

[14] K. Kang, B. C. Paul, and K. Roy. Statistical timing analysis using levelized covariance propagation considering systematic and random variations of process parameters. In *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, volume 11, pages 848–879, October 2006.

[15] K.Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan. Temperature-aware microarchitecture. In *the 30th International Synposium on Computer Architecture*, pages 2–13, 2003.

[16] J. C. Ku, S. Ozdemir, G. Memik, and Y. Ismail. Thermal management of on-chip caches through power density minimization. In *ACM/IEEE 38th International Symposium on Microarchitecture (MICRO'05)*, 2005.

[17] K. meng and R. Joseph. Process variation aware cache leakage management. In *Proceedings of the 2006 International Symposium on Low Power Electronics and Design*, pages 262–267, 2006.

[18] S. Mukhopadhyay, H. Mahmoodi, and K. Roy. Modeling and estimation of failure probability due to parameter variations in nano-scale srams for yield enhancement. In *VLSI Circuit Symposium*, 2004.

[19] Predictive technology model. http://www.eas.asu.edu/ ptm/.

[20] K. B. R. Rao, A. Srivastava, and D. Sylvester. Statistical analysis of subthreshold leakage current for vlsi circuits. In *IEEE Trans. on VLSI Systems*, volume 12, pages 131–139, Feb 2004.

[21] K. Sankaranarayanan, S. Velusamy, M. Stan, and K. Skadron. A case for thermal-aware floorplanning at the microarchitectural level. In *Journal of Instruction-Level Parallelism*, Sept 2005.

[22] T. Sherwood, E. Perelman, and B. Calder. Basic block distribution analysis to find periodic behavior and simulation points in applications. In *Proc. PACT*, 2001.

[23] P. Shivakumar and N. P. Jouppi. Cacti 3.0: An integrated cache timing, power, and area model, 2001.

[24] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers. Exploiting structural duplication for lifetime reliability enhancement. In *ISCA*, pages 520–531, 2005.