PV-aware Replacement Policy for Two-level Shared Cache

Bindu Agarwalla
Dept. of CSE, IIIT Guwahati
Guwahati, Assam, India - 781015
bindu.agarwl@gmail.com

Nilkanta Sahu
Dept. of CSE, IIIT Guwahati
Guwahati, Assam, India - 781015
nilkanta@iiitg.ac.in

Shirshendu Das
Dept. of CSE, IIT Ropar
Ropar, Punjab, India - 140001
shirshendu@iitrpr.ac.in

2022 IEEE International Symposium on Smart Electronic Systems (iSES) | 979-8-3503-9922-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/ISES54909.2022.00100

***Abstract*—The performance of modern multicore processes largely depends on the performance of their shared Last Level Cache (LLC). A larger LLC can reduce misses while increasing access latency. Tile-based chipmultiprocessors (TCMP) are also proposed with two levels of shared caches to handle such situations. Among the two levels of shared caches, the bottom level is designed with DRAM technology for better capacity, and the upper level is designed with SRAM technology for better latency. Because it is in the last level, the LLC is the DRAM-based shared cache. Both the shared caches are normally divided into multiple banks. The core in such TCMPs can have its own private cache above these two levels of shared caches. Process variation has been observed to have a much greater impact on DRAM-based memory chips than on SRAM-based memories. Hence, the banks of the DRAM-based LLC may not behave uniformly. Some of the banks may experience high energy costs and access latency because of PV. In this work, we have proposed a replacement policy for the SRAM-based shared cache (upper-level shared cache), which reduces the requirements to access the PV-affected LLC banks significantly. This is achieved by relaxing the blocks of the affected LLC banks to remain in the upper-level shared cache (SRAM-based) for a longer time. The experimental analysis found that the proposed replacement policy improves performance by 13%, 10%, and 6% as compared to three existing replacement policies.**

***Index Terms*—DRAM Last Level Cache, Process Variation, Replacement Policy.**

I. Introduction

The performance of the Last Level Cache (LLC) in modern Chipmultiprocessors (CMP) is crucial for the better performance of the system. Normally, in LLC, capacity is more important than latency. Hence, efforts are going on to design a larger-sized LLC with less latency and energy overhead. Traditionally, SRAM technologies are used to design cache memories. Though the SRAM technology is faster, its low density makes it challenging to use it for designing larger-sized LLCs. DRAM technology is slower than SRAM but it has high density. Hence, using DRAM technology to design the LLC is an alternative choice [1]. However, the DRAM has some disadvantages, like refresh overhead, row hammer attack, etc. Research is going on to reduce such overheads. The LLC designed with DRAM is called DRAM LLC.

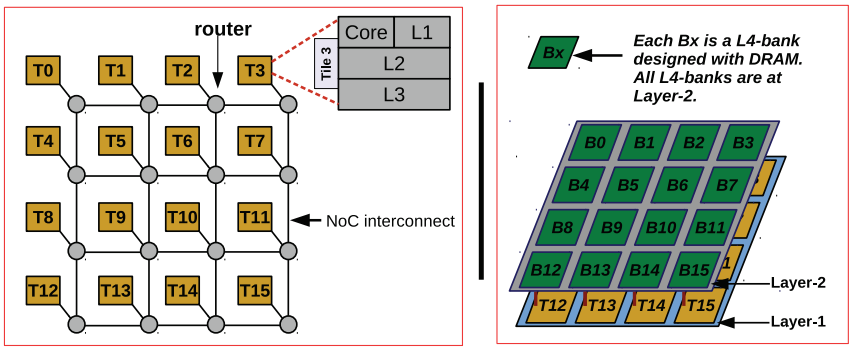
This work is partially supported by Science and Engineering Research Board (SERB) grant, numbered ECR/2017/000749, funded by the Department of Science & Technology, Government of India.

Tiled-based CMP (TCMP) is one of the most commonly used CMP structures. In TCMP, there are multiple tiles connected with some on-chip interconnects. Each tile has a core and its own upper-level private cache. A part of the shared LLC (bank) is also attached to each tile. Figure 1(a) shows the structure of TCMP. Figure 1(b) shows a tiled-based CMP (TCMP) with DRAM LLC. This is a 3D TCMP where there are two levels of shared caches: L3 and L4. The design is proposed in [2], [3]. In this 3D TCMP, there are two layers. In Layer 1, the tiles, private caches (L1 and L2), and a shared L3 cache are present. All the caches in Layer-1 are designed with SRAM. Layer 2 is placed vertically above Layer 1. This layer only has a large-sized LLC (L4) designed with DRAM. The prior work already showed the efficiency of such 3D TCMPs over other TCMPS having only SRAM-based LLCs. A block, based on its bank index bit, always maps to the same bank in L3 and L4. The bank in which the block b maps is called the *home-bank* of the block. A block always has one home-bank in L3 and one home-bank in L4. In this work, we have considered 3D TCMP (shown in Figure 1) as the baseline.

Because of different fabrication complexities, all regions of a chip may not behave uniformly. This phenomenon is called "process variation" (PV) [4] It has been found that the PV impact is greater on DRAM-based LLCs than on SRAM-based LLCs. Because of PV, all banks in L4 (L4-bank) may not have similar behavior. Some of these L4 banks may have more latency and energy consumption. The L4 banks affected by PV are called PV-affected banks or unhealthy banks. Unhealthy L4 banks can also be categorized based on the impact of PV. Two unhealthy L4 banks may belong to two different categories if the latency and energy overhead is not similar. The existing work tries to mitigate the PV impact in the LLC by shutting down the PV-affected part of the LLC. However, it may not always be possible to shut down the PV-affected L4 banks if the LLC needs more space to maintain performance. In such cases, the LLC resizing techniques as proposed in [2], [3] cannot be applied. However, the impact of PV in the LLC can be reduced if somehow the number of accesses to the PV-affected L4 banks can be reduced. In this work, we have proposed a PV-aware replacement policy (PVaR) for the L3 cache. PVaR helps L3 to keep the important blocks of the PV-affected L4 banks for a longer time in L3. Here, the blocks of PV-affected banks mean the blocks whose home bank in L4

2022 IEEE International Symposium on Smart Electronic Systems (iSES)

979-8-3503-9922-6/22/$31.00 ©2022 IEEE
DOI 10.1109/iSES54909.2022.00100



(a) Layer-1 of 3D TCMP. (b) The 3D TCMP.

Fig. 1. Example of 3D TCMP [3].

is PV-affected.

A replacement policy determines the victim block, out of all possible candidates, that should be removed from the cache upon the arrival of a new block. To address this issue, we can use a well-behaved replacement policy in the L3 shared cache. For most of the replacement policies, the primary goal is to reduce the number of misses in the cache, and the victim selection process is based on the recency or frequency of usage. In our memory hierarchy, we have a DRAM cache as the LLC, that needs to serve the misses caused in the L3 SRAM shared cache. As our LLC is PV-affected, all the banks are not in the same health condition. Due to varying health conditions, the banks will not be able to serve the misses in a uniform way. Some banks may take more time and energy compared to all others. The health of a block is the health of the bank it belongs to.

Suppose whatever block is being evicted belongs to a PV-affected L4-bank and has higher re-usability in the future, then we need to again bring the block from the PV-affected L4-bank to the L3 cache. This will lead to higher latency and energy consumption. As our main goal is to reduce the energy consumption while using the DRAM LLC, a smart replacement policy in the L3 SRAM shared cache is proposed to reduce the overall energy consumption and access latency by accessing PV-affected DRAM banks fewer times and by reducing the number of misses in the L3 cache. Our replacement policy selects a victim block based on its predicted reuse interval, called the Predicted Re-Reference Value (RRPV) [5] and the health condition of its home bank in the L4 cache. The experimental result shows that our proposed policy shows better performance than all the existing replacement policies. The improvements are 13%, 10% and 6% over SRRIP [5] , DRRIP [5], and HawkEye [6] respectively.

The rest of the paper is organized as follows. The background information related to this work is discussed in Section II. The proposed idea is discussed in Section III. Section IV discusses the experimental details of the work. Finally, the work concludes in Section V.

II. Background

A. Set-Associative Cache and Replacement Policies

A set-associative cache has multiple sets and each set has a fixed number of ways. An N-way set associative cache means each set has a N number of ways. The set-associative caches are divided into tag-array and data-array. Each entry in the data-array can store one cache block. The corresponding entry in the tag array stores the metadata of the block. Here, metadata means tag address, valid bit, dirty bit, and sharers bits, etc. A block is always mapped to a fixed set based on the set-index of the block address. Figure 2 shows an 8-way set associative cache. While fetching a new block in the cache, if the mapped set is full, an existing block from the set needs to be evicted first to place the new block. This is called the replacement policy of the cache. Each set in the cache has its own replacement policy. Any replacement policy has three major modules: insertion, promotion, and eviction. The insertion modules decide where to put a block when first fetched from a lower level of memory. For example, in the case of the LRU replacement policy, the newly fetched blocks are marked as MRU. The promotion policy means to promote a block after getting a hit in the cache. In LRU policy, a block gets promoted to MRU after getting hit in the cache. The eviction module decides which block should be evicted from the cache for placing the new block. The process is also called the victim selection process. In LRU policy, the victim selection process always selects the LRU block as the victim block.

This work is based on a replacement policy called RRIP [5]. In RRIP, each block maintains an RRPV value to calculate the victim block. Some other well-known replacement policies are [6]–[8]. In RRIP a block gets RRPV as $2^M - 2$ during insertion. The block gets promoted to $RRPV = 0$ for every hit. The victim selection process selects the block having $RRPV = 2^M - 1$ as a victim. In case, if no block in the set has $RRPV$ as $2^M - 1$ then all the RRPV values in the set are incremented until one of the RRPV becomes equal to $2^M - 1$. The value of RRPV considered in [5] is $log_2(N)$, where N is the associativity of the cache.

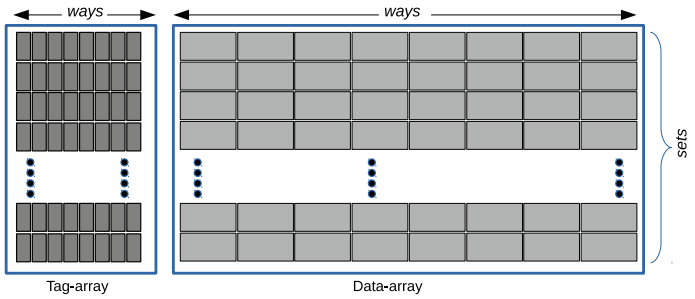


Fig. 2. Example of an 8-way set associative cache. The set-associative caches are divided into tag-array and data-array. The associativity and sets remain the same in both array.

B. Access Patterns of Applications

It is observed that during the execution of an application, various access patterns are followed. And no single replacement algorithm is suitable for all access patterns. Let b_i denotes the address of a cache line and $\{b_1, ..., b_k\}$ denotes a temporal sequence of references to k unique addresses. Let $P\varepsilon\{b_1, ..., b_k\}$ denotes a temporal sequence that occurs with some probability ε. A temporal sequence that repeats N times is represented as $\{b_1, ..., b_k\}^N$. The cache access patterns can be classified into the following categories:

- Recency-friendly Access Patterns: In recency-friendly access patterns, if b_i is referenced at time t_i, then in the future also it is going to be referenced again. Such an access pattern has a near-immediate re-reference interval. So, these access pattern gets the benefit from the LRU replacement as this algorithm favors those that are most recently accessed compared to whatever was referred in the past.
- Thrashing Access Patterns: In thrashing access patterns, a cyclic access pattern of length k is repeated N times. Thrashing occurs when the cache cannot accommodate the current working set. When k is less than or equal to the number of blocks in the cache, the working set fits into the cache. For such access patterns, LRU does not provide cache hits. When the available cache is less than k entries, the optimal replacement policy preserves some of the working set in the cache.
- Streaming Access Patterns: Streaming access patterns are the ones having workloads with infinite re-reference intervals. Consequently, streaming access patterns receive no cache hits under any replacement policy. These types of access patterns are also called scanning access patterns.
- Mixed Access Patterns: Mixed access patterns can be characterized as workloads where some references have a near immediate re-reference interval while other references have a distant re-reference interval.

C. Process Variation

As a result of poor device specification control during the manufacturing, process variation (PV) develops. cite [9]. Gate length, wire width, threshold voltage, and resistance per unit length are all variables that might be affected. The performance, stability, and leakage of integrated circuits are profoundly impacted by PVs. In 3D DRAM, PVs are taken into account in the access transistor, capacitor, and peripheral circuits [10]. There is the possibility of both systematic and random PV in transistors. Lithographic aberrations, which induce a shift in the effective gate length L_{eff}, are a common source of these systematic shifts. We should expect smaller variances between devices that are close together and larger variations between devices that are further away because of the significant spatial connections between these differences. Changes in the threshold voltage due to random fluctuations are not predictable and do occur. V_{th}. Systematic shifts will cause shifts in effective gate length (L_{eff}). As a result, all the word lines in a given bank will require distinct activation durations as one moves from the top to the bottom. When the value of L_{eff} or V_{th} varies, the leakage in the circuit changes as well. Refreshing a leaky DRAM sub-bank often decreases performance and increases energy consumption.

D. Motivation of the Work

The replacement policies mentioned in Section II-A do not consider the PV impacts of lower-level memories. The 3D TCMP as shown in Figure 1 has two levels of shared memories L3 and L4. In such TCMPs, the chances of being affected by PV are higher in L4. Assuming that the impact of PV is not uniform across L4, some of the L4 banks are heavily affected by PV while others may be normal. Accessing those PV-affected L4 banks is expensive in terms of both latency and energy. The replacement policy of L3, if aware of the health condition of L4 banks, may reduce the number of accesses to those L4 banks. This is possible by retaining the important blocks of the affected L4 banks for a longer time in L3. As per our knowledge, none of the existing replacement policies is designed to consider the PV impacts of lower-level cache memory. Hence, in this work, we have been motivated to propose a PV-aware replacement policy for the upper-level shared cache (L3). The replacement policy handles the PV impacts of the lower-level shared cache (L4).

III. Our Proposed Replacement Policy

The 3D TCMP shown in Figure 1 is considered the baseline of this work. We propose a smart cache replacement policy, PVaR, for the shared L3 cache. In general, the main goal of a cache replacement policy is to reduce the number of misses. But our smart cache replacement policy not only reduces the number of misses but, on top of that, minimizes the access to the PV-affected L4 banks, reducing the overall energy consumption. As PV-affected L4 banks are slow and power-hungry banks, reducing access to these L4 banks reduces the access time as well as the energy consumption of the cache memory. Due to PV, the health condition of an L4-bank can be affected. The health condition of a bank is defined using two parameters: latency and energy consumption. The health of a block is modeled using 2 bits of the health index, whose possible values are: $3, 2, 1$, and 0. Based on these values, the L4 banks can be grouped into multiple categories. The value of 3 means the most unhealthy L4 banks. These L4 banks are affected the most by PV. The value 0 means, healthy blocks, which are not being affected by PV.

The variation in latency and energy consumption of the L4 cache due to PV is calculated as per the procedure discussed in [3]. The procedure for calculating the energy consumption and access latency of each L4-bank is also taken from [3]. The health index value of an L4-bank is assigned to 0, when the value of both the parameters (Energy and Latency) is less than 25% of the maximum value. The L4-bank consuming the highest energy/latency is considered as the maximum value. When one of the parameters is greater than 25% and less than 50% of the maximum value, the health index assigned is 1, and when both the parameters are greater than 25% and less than 50% of the maximum value, the health index assigned is 2.

The proposed replacement policy (PVaR) gives relaxations in choosing a block as a victim in L3, whose home-bank in L4 is PV-affected. The policy selects a victim block based on its predicted reuse interval, called the Predicted Re-Reference Value (RRPV), and the health condition of its home bank in the L4 cache. Similar to SRRIP, with every block we have associated an M-bit RRPV. Here M is normally the bit required to represent the associativity of the L3 cache. With M bits the maximum RRPV value possible in $2^M - 1$ and the minimum value possible is 0. The health of each L4-bank is observed periodically and it will be stored in each L3 cache. Hence, each L3 cache has knowledge of the health issues of all the L4 banks. The home-bank of any block in L3 can be easily calculated from the bank-index bits of the block address. In the rest of the discussion, we consider that a block b belongs to an L4-bank Q, if Q is the home-bank of b at L4.

Each block in the L3 is also associated with another counter called *chance-counter*. The chance-counter stores how many extra chances should be given to the blocks from unhealthy L4 banks to remain in the L3 cache. The minimum value is 1 and the maximum value is 3. Hence, chance-counter needs two bits for each block. If the value of the chance-counter for a block is 2, then the block gets two chances before being selected as a victim. The different modules of the replacement policy will be discussed next.

A. Insertion, Promotion, and Eviction of PVaR

In this section, the three important modules of PVaR are discussed. These modules are insertion, promotion, and eviction. Any replacement policy, as discussed in Section II-A, must have these three modules.

1) Eviction Policy:

A) Select the block having $RRPV = 2^M - 1$ as the victim block. In case of multiple such blocks, the victim block is selected randomly.
B) If there are no blocks having $RRPV = 2^M - 1$ then do the following for all the blocks in the set:
   i) If the RRPV of a block is $2^M - 2$ and its chance-count is not zero, decrement the chance-count but do not increment the corresponding RRPV value.
   ii) Increment the RRPV value in any other case not satisfying the condition mentioned in B(i).
C) Go to step A.

The policy always selects victims having RRPV as $2^M - 1$. However, the blocks from unhealthy L4 banks take more time to get incremented from RRPV values $2^M - 2$ to $2^M - 1$ because of the chance-count. As per the condition mentioned in Rule B(i), a block belonging to an unhealthy L4-bank cannot be incremented farther than $2^M - 2$ unless its chance-counter gets zero. Hence, based on the initial value of the chance-counter, the blocks belonging to different unhealthy groups of L4 banks get more relaxation to remain in the L3 cache. Rule A and B make sure that even in the case of all the blocks in the L3 set coming from the same unhealthy L4-bank, the replacement policy will not get stuck. The blocks from more unhealthy L4-bank get higher chances to remain in L3. Step A(i) ensures that no block is allowed to stay in the cache infinitely. If the block is a dead block, it will eventually be evicted from L3 no matter which health category it belongs to.

2) Insertion Policy: Insert a block as $RRPV = 2^M - 2$ same as RRIP [5]. The PVaR also sets the chance-count value of the block at the time of insertion. The rules for initializing the chance-count value are as follows:

A) If the block belongs to L4-bank having health-index 0, set chance-count as 00.
B) If the block belongs to L4-bank having health-index 01, set chance-count as 01.
C) If the block belongs to L4-bank having health-index 02, set chance-count randomly to either 01 or 02.
D) If the block belongs to L4-bank having health-index 03, set chance-count randomly to either 01, 02, or 03.

The random value assigned in cases C and D is because not all blocks from these categories are reused in the near future. Giving a higher chance to a block that has never been reused in the near future has no advantage. Removing such blocks early does not increase the number of accesses in the L4 cache. Also, L3 can be used for some other important blocks.

Specification	Values
Number of tiles (cores)	16, 2 GHz each.
Architecture used	Alpha
Level of cache memory	4 (L1/L2 Private, L3/L4 shared)
L4 cache size/assoc/banks	32MB/29-way/16.
L3 cache size/assoc/banks	8MB/8-way/16
L2 cache size/assoc/nums	256KB/4-way/16
L1 cache size/assoc/nums	64KB/2-way/16
Cache block size	64 Bytes
NoC	Mesh based NOC with TSV
Main memory size	4GB

TABLE I
3D TCMP SPECIFICATIONS. HERE "NUMS" MEANS NUMBER. THE 29-WAY ASSOCIATIVE L4 CACHE IS AS PER [11].

3) Promotion Policy: On every hit of the block in L3, promote the block as RRPV 0. Also initialize the chance-counter as per the rules mentioned in Section III-A2.

B. Summary of PVaR

PVaR gives extra chances for the blocks from unhealthy L4 banks to remain in the L3 cache for a longer time. However, if the block does not get a hit in the near future, it will eventually waste all its chances. After wasting all its chances, the RRPV value of such blocks reaches $2^M - 1$. Hence eventually evicted from the L3 cache. In the case of promotion during this time, the block gets new chances and is also promoted to RRPV as 0. Note that the chance-count of a block is initialized twice. First at the insertion of the blocks and second during every hit (at L3) in the block. The chance-count value is only allowed to decrement by Rule B(i) of Section III-A1.

IV. EXPERIMENTAL ANALYSIS

The full system simulator called gem5 [12] is used for the experimental analysis. As mentioned before in this work we have considered 3D TCMP as our baseline architecture. The 3D TCMP is mentioned in Section I and Figure 1. The support of DRAM memory in gem5 and four level MESI_CMP protocol required for 3D TCMP are implemented in [2]. In this work we have used the baseline model implemented previously in [2]. The complete system parameters are shown in Table III-B. Ten Parsec Benchmarks [13] have been used for this experiment. The benchmarks are *bodytrack*, *canneal*, *dedup*, *facesim*, *ferret*, *fluidanimate*, *rtview*, *swaptions*, *vips* and *x264*. All the benchmarks are executed in full system mode for 2 billion instructions after entering the ROI (Region of Interest) or till the ending of the ROI. The number of threads executed for each benchmark is 16. After starting the ROI, we warm up the cache for 1 million instructions and then started the actual simulation.

We have compared our policy with recent replacement policies: LRU, SRRIP [5], DRRIP [5], and HawkEye [6]. As per our knowledge, no replacement policy has been proposed considering the PV affects on its lower-level memory. Hence the proposed work is compared with the existing techniques where no PV affects were considered. The same 3D TCMP architecture is used as the baseline for all the techniques. As

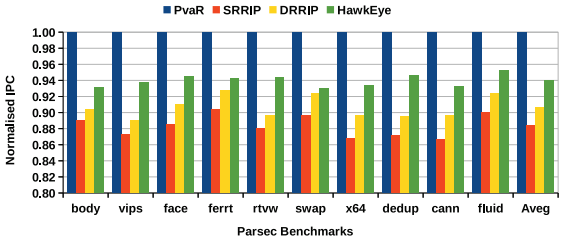


Fig. 3. IPC comparison of the proposed policy with existing techniques.

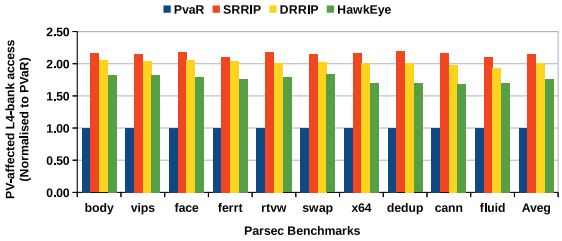


Fig. 4. PV-affected LLC bank access comparison of the proposed policy with existing techniques.

discussed in Section III the replacement policy is proposed for the L3 cache which is shared and placed at the upper level of the LLC (L4). The structure of LLC (i.e., L4) as well as the entries 3D TCMP is already discussed in Section III. For LLC, the LRU policy is used for all the experiments. This is also true for the two upper-level caches L1 and L2. The changes are in the L3 cache. The varying latency and energy consumption of L4 banks due to PV is considered for all executions. The PV variations are calculated as per the method discussed in [3]. The cache modeler tool CACTI [14] is used to calculate the energy consumption and latency of the cache memories.

A. Performance and Energy Consumption Analysis

Figure 3 shows the IPC comparison of the proposed PVaR with SRRIP, DRRIP and HawkEye. It can be observed from the figure that our proposed policy shows better performance than all the existing replacement policies. The improvements are 13%, 10%, and 6% over SRRIP, DRRIP, and HawkEye respectively. The main reason of this improvement is to reduce the number of accesses to the PV-affected LLC banks. To verify our claim Figure 4 shows the reduction in the number of accesses to PV-affected banks. The figure shows that the proposed replacement policy has the lowest number of accesses to the PV-affected banks. However, the number of accesses to the healthy LLC banks is slightly increased in our proposed policy. These increase is not degrading the performance because: (a) The number of high latency misses reduced by the proposed policy is much higher. (b) Every block from unhealthy blocks cannot get higher chances to remain in the L3 (as per Rule B(i) of Section III-A1). Hence

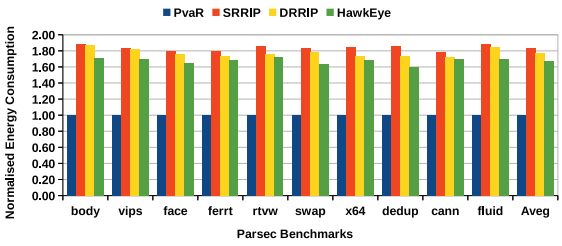


Fig. 5. Energy consumption comparison of the proposed policy with existing techniques.

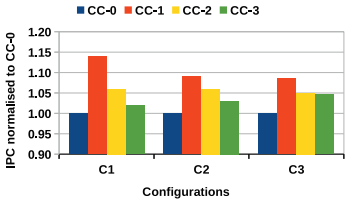


Fig. 6. Performance of different cache configurations with different chance-count values. Here CC-x means chance-count value as x.

the dead blocks from such unhealthy L4 banks cannot stay a long time in L3.

Figure 5 shows the reduction in dynamic energy consumption by our proposed policy as compared to the existing techniques. It can be observed from the figure that the dynamic energy reductions are 46%, 44%, and 40% as compared to the SRRIP, DRRIP, and HawkEye. The main reason for such a reduction is the reduction of access to the PV-affected banks. Accessing a PV-affected bank consumes significantly higher energy than a healthy bank. Since we are not shutting down any components like [3], the static energy consumption remains the same.

B. Sensitivity Analysis

As per the rule mentioned in Section III-A2, the blocks from the highly affected L4 banks (health index 2 or 3) are getting a random chance-count in L3 cache. The minimum chance-count getting by the blocks from these L4 banks is 1. However, the maximum value is 2 and 3, respectively, for health indexes 2 and 3. Figure 6 shows the IPC comparison for different configurations: C1, C2, and C3. In configuration C1, all the L4 banks are either normal or PV-affected, having a health index of 1. In C2, the L4 banks are either normal or PV-affected with health index 2, and in the case of C3, the L4 banks are either normal or PV-affected with health-index 3. For all configurations, the experiments are run for different chance-count values for the blocks coming to L3 from the affected banks of L4. From the figure, it can be observed that in the case of C1, chance-count 1, gives more benefits than the other chance-count options. For C2, and C3, the improvements are slightly better for chance-count 1, but as the chance-count value increases, the IPC decreases. This is because the dead blocks and never reused blocks from such L4 banks stay in the L3 for a longer time. Hence, in Section III-A2, we have used a random chance-count number for such L4 banks.

V. Conclusion

In this paper, we have proposed an efficient replacement policy (PVaR) for the L3 caches of 3D TCMP. In 3D TCMP, the last two levels of caches (L3 and L4) are shared. L3 is designed with SRAM and L4 is designed with DRAM. The L4 is also the LLC of the 3D TCMP. The PVaR is applied to the banks of L3 for handling the PV-affected in the L4 cache memory of the 3D TCMPs. Because of the PV impact, some of the L4 banks experience high energy consumption and access latency. The PVaR keeps the important blocks of the PV-affected L4 banks for longer periods of time, such that the number of accesses to such L4 banks can be reduced. Hence, the overall system performance and energy consumption are enhanced. Experimental analysis shows that the proposed replacement policy performs better than some existing replacement policies.

References

[1] S. Mittal and J. S. Vetter, "A survey of techniques for architecting dram caches," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 6, pp. 1852–1863, June 2016.

[2] "Efficient cache resizing policy for dram-based llcs in chipmultiprocessors," *Journal of Systems Architecture*, vol. 113, p. 101886, 2021.

[3] B. Agarwalla, S. Das, and N. Sahu, "Process variation aware dram-cache resizing," *Journal of Systems Architecture*, vol. 123, p. 102364, 2022.

[4] B. Zhao, Y. Du, J. Yang, and Y. Zhang, "Process variation-aware nonuniform cache management in a 3d die-stacked multicore processor," *IEEE Transactions on Computers*, vol. 62, no. 11, pp. 2252–2265, Nov 2013.

[5] A. Jaleel, K. B. Theobald, S. C. Steely Jr, and J. Emer, "High performance cache replacement using re-reference interval prediction (rrip)," *ACM SIGARCH computer architecture news*, vol. 38, no. 3, pp. 60–71, 2010.

[6] A. Jain and C. Lin, "Back to the future: Leveraging belady's algorithm for improved cache replacement," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2016, pp. 78–89.

[7] I. Shah, A. Jain, and C. Lin, "Effective mimicry of belady's min policy," in *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2022, pp. 558–572.

[8] K. K. Dutta, P. N. Tanksale, and S. Das, "A fairness conscious cache replacement policy for last level cache," in *2021 Design, Automation and Test in Europe Conference and Exhibition (DATE)*, 2021, pp. 695–700.

[9] H. Kim, I. Choi, J. Lim, H. Oh, and S. Kang, "Process variation-aware bridge fault analysis," in *2016 International SoC Design Conference (ISOCC)*, Oct 2016, pp. 147–148.

[10] M. Mutyam and V. Narayanan, "Working with process variation aware caches," in *2007 Design, Automation Test in Europe Conference Exhibition*, April 2007, pp. 1–6.

[11] F. Hameed, L. Bauer, and J. Henkel, "Architecting on-chip dram cache for simultaneous miss rate and latency reduction," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 4, pp. 651–664, 2015.

[12] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti *et al.*, "The gem5 simulator," *ACM SIGARCH Computer Architecture News*, vol. 39, no. 2, pp. 1–7, 2011.

[13] C. Bienia, "Benchmarking modern multiprocessors," Ph.D. dissertation, Princeton University, January 2011. [Online]. Available: http://parsec.cs.princeton.edu/

[14] N. Muralimanohar, R. Balasubramonian, and N. Jouppi, "Optimizing nuca organizations and wiring alternatives for large caches with cacti 6.0," in *Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO 40, 2007, pp. 3–14.