Coarse Granularity Data Migration Based Power Management Mechanism for 3D DRAM Cache

Litiao Qiu(✉), Lei Wang, Hongguang Zhang, Zhenyu Zhao, and Qiang Dou

School of Computer, National University of Defense Technology,
Changsha 410073, Hunan, China
{qiulitiao,hongg_z}@163.com, {leiwang,zyzhao}@nudt.edu.cn,
douq@vip.sina.com

Abstract. 3D-stacked technology is a promising solution to improve the performance of on-chip memory system. In our work, a 3D DRAM Cache with high density and wide bandwidth is utilized as the Last Level Cache (LLC). With the same Cache area, a 3D DRAM Cache shows superior capacity, bandwidth, cost performance ratio to a SRAM Cache. However, 3D DRAM storage has a problem of high power consumption. The power consumption of die-stacked DRAM is 5x compared to plane DRAM. In order to solve this problem, we proposed a power management mechanism for 3D DRAM Cache in this paper. The core idea of our mechanism is closing the infrequent accessed banks for saving power consumption. We design and implement a trace-driven 3D DRAM Cache simulator based on DRAMSim2. Experiment result of SPEC CPU2006 shown that for most applications, some banks have little access during execution. We proposed a coarse granularity data migration based power management mechanism. Compared with the system without power management mechanism, the static power consumption of some application decreased to 0.75x, a portion of application reach to 0.375x.

Keywords: 3D-stacked · DRAM cache · Memory system · Power management · LLC

1 Introduction

3D stacking technology applies through-silicon-vias (TSV) to connect different dies. Thus, in the same area, 3D DRAM has a bigger capacity and wider bandwidth compares to a 2D DRAM. However, 3D DRAM has the problem of high power consumption. In Fig. 2, power consumption of 3D DRAM is 5x larger than 2D DRAM.

Based on the comparison results between 2D DRAM and 3D DRAM, although 3D DRAM cache has a smaller area and bigger capacity, we cannot ignore the huge power consumption caused by 3D DRAM.

This work was supported by the National Nature Science Foundation of China (61402501, 61272139).

J. Wu and L. Li (Eds.): ACA 2016, CCIS 626, pp. 15–27, 2016.
DOI: 10.1007/978-981-10-2209-8_2

In recent works, many researchers proposed various way to manage the power of on-chip memory. They focus their attention on the different cause of power consumption, dynamic power and leakage power. Some studies dealt with the hit rate of Cache, the higher the hit rate, the more power can be saved. Other researchers [4] tried to redesign the framework of Cache, they separate the data array of the Cache into sub-banks. Only activate the bank when the corresponding data is needed. In addition to that, in order to save the dynamic power, activate less tag bits and data in an access is a method adopted by [5]. They access the tag array in the first phase. In the second phase, only the corresponding tag bits are accessed, can the data referenced. However, in a big capacity LLC, leakage power accounts a big portion in whole power consumption. Coarse granularity execution is practical enough to figure out the power consumption problem of 3D DRAM Cache.

In this paper, we proposed a Coarse Granularity Data Migration power management mechanism. Based on the access pattern of the application, we choose the unlikely accessed banks to be powered-off. Thus, we design and implement a simulation platform which has a core, 3-level Cache (LLC is a 3D DRAM Cache) and 2D main memory. Experiment result with SPEC CPU2006 shown that in most applications, some banks have little access during execution. Thus, the static power consumption of some application decreased to 0.75x, a portion of application reach to 0.375x.

The organization of the rest of the paper is as follows: In Sect. 2, we demonstrate the background and motivation of our work. The main idea of power management in 3D DRAM Cache is shown in Sect. 3. Section 4 gives a detail analysis on experimental setup and results of our mechanism. Section 5 presents a brief review of power management and 3D-stacked system. Section 6 is the conclusion of the paper.

2 Background and Motivation

Recent works proposed many DRAM-cache prototypes to address the memory bandwidth and latency wall in Last Level cache. Hit ratio, hit latency and tag overhead determine the challenges of implementing a DRAM Cache.

In order to reduce the miss rate, a cached DRAM [1] integrates SRAM cache in the DRAM memory to exploit its locality in memory accesses and storage efficiency. In such DRAM caches, SRAM tags are placed in the stacked DRAM along with the data blocks [1, 2]. However, this can potentially require two DRAM accesses per cache lookup (one for the tag look up and one for data). Thus, in the worst case, it costs double access latency.

A state-of-art DRAM cache method, Alloy Cache [2], organizes as a direct-mapped Cache, which is optimized for latency. Although reduce the hit rate, improving the cache access latency greatly. The DRAM cache model used in our work is inspired by Alloy Cache, which bursts tag and data in a single stream to wipe out the tag serialization delay.

Compare to other DRAM caches, Alloy cache has no SRAM tag overhead, low hit latency and scalability. For a die-stacked DRAM Cache, we can get high effective capacity and high hit rate as well. Thus, we exploit a die-stacked DRAM cache as our experimental model.

In our work, Alloy Cache is used as DRAM cache prototype. Our L3 DRAM cache saves two kinds of information, tag and data. Alloy Cache is an effective latency-optimized Cache architecture. It alloys or combines the tag and data into one basic unit (Tag and Data, TAD), instead of separating cache constructions into two parts (tag store and data store). For Alloy cache streams tag and data in one burst, it can get rid of the delay because of tag serialization. It helps handle cache misses faster without wasting time to detect cache miss in the same situation.

In Fig. 1, each TAD represents one set of the direct-mapped Alloy Cache. In our DRAM cache, every data line has a tag. The address is compared with the tags in DRAM cache. If it is the same, then hit. Or vice versa. In our paper, for a physical address space of 64 bits, 41 tag bits are needed. The minimum size of a TAD is thus 72 bytes (64 bytes for data line and 8 bytes for tag). There are 28 lines in a row of an Alloy Cache.

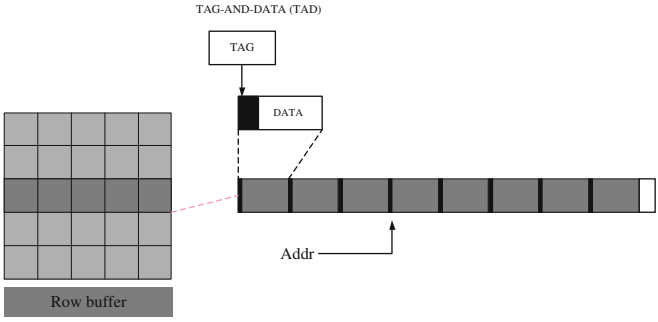

Fig. 1. Structure of alloy cache

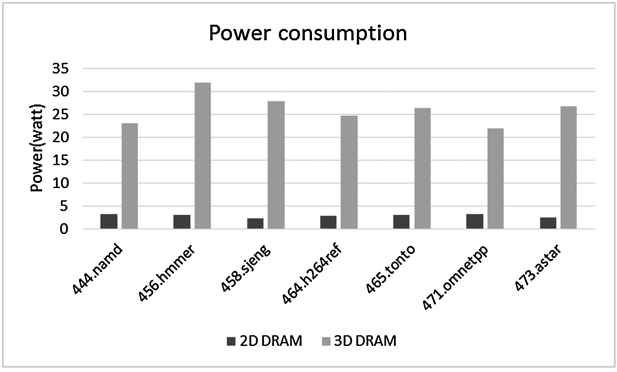

Fig. 2. Power consumption in baseline and 3D DRAM cache system with no power management

We conduct some experiments on power consumption of die-stacked DRAM. We run 200 billion instructions for each application. Then, using the trace from Gem5 feed to DRAMSim2. Comparing to a 2D DRAM, it demonstrates big leakage power consumption in 3D DRAM. The result is shown in Fig. 2.

3D-stacked framework shows great merits in capacity and bandwidth. However, as is shown in Fig. 2, power consumption is a big problem in improving processor performance. For 3D DRAM, the power consumption is 5x larger than a 2D DRAM in average. Thus, the main purpose of our research is to maintain the big capacity of the L3 die-stacked DRAM Cache, while reduce power consumption as well.

3 Main Idea

The main idea of Coarse Granularity Data Migration based power management mechanism is closing the infrequent accessed banks of 3D DRAM Cache to save power consumption. How to find the bank that should be closed and how to migrate the data in these banks are the challenges that we must solve. An offline method to figure out the access pattern of applications is used in this paper. Figure 3 illustrates the flow of our Coarse Granularity Data Migration mechanism.

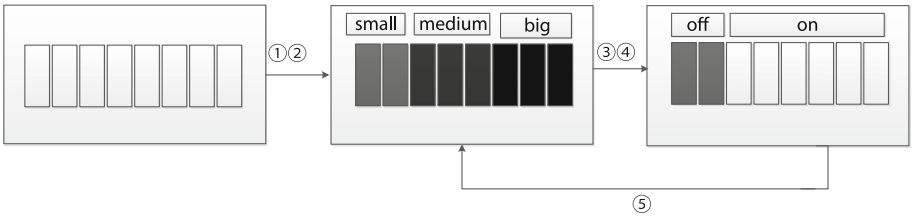

Fig. 3. The flow of coarse granularity data migration method. (the flow is explained below)

Bank closing steps. ①Analyze the access pattern of the benchmarks. Cluster the banks into three groups according to the references in banks. ②Name these three groups as Cluster small, medium and big. ③Migrate the data to other banks and close the last two banks. ④After the bank power-down, remapping the references originally mapping to the closed bank. ⑤After every interval time, check the access pattern again. If it changes, open the closed bank and run this mechanism over again.

3.1 Access Pattern Analysis

For exploring the possibility of making full use of the banks in DRAM cache, we analyze the access pattern of some programs in SPEC CPU2006. The result is demonstrated in Fig. 4, x-axis represents the bank ID, y-axis is the rows. The rows are separated into 8 parts, every part has 1,024 rows. The right column shows how color reflects access number. In these hot spot figures, deeper the color, more references in that area.

In Fig. 4, except 437, we can see obvious cluster in each application. For example, references concentrate in bank 0 and bank1 in 456.hmmer. While in 444.namd, there is no access in bank 5, 6 and 7. It means some of the banks in an application are not fully utilized. These infrequent used banks can be closed during its runtime.

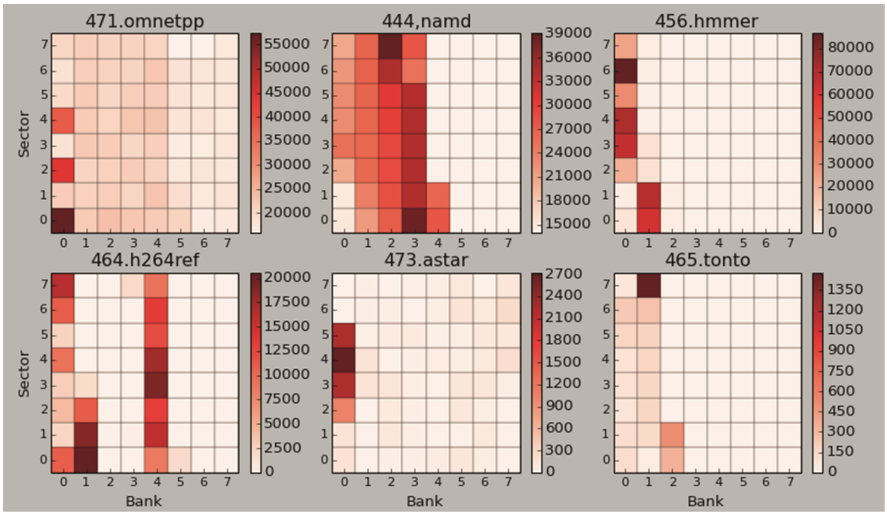

Fig. 4. Access pattern of different applications

3.2 Cluster Banks Using K-Means

In order to reduce power consumption, we can power down those unusually used banks. Thus, we exploit K-means to find out those banks have little references. Banks are separated into three clusters.

First of all, pick three access number of banks as centroids randomly. Then, it calculates the distance from other access number to the centroid. The banks have the closing access number with one centroid will be allocated to the same cluster. The next step is calculating the average value of each cluster, regard it as the new centroid. Thus, we have three new centroids. Calculating the distance between the access number in all banks and the update centroids. We get three new clusters. Then iterating the process continuously. Until we get the final three clusters of banks.

In our work, we have 8 banks. With the purpose of closing the unlikely accessed banks. We divide these 8 banks into three groups. In Cluster big, banks get the most of the access. So we leave it alone. Our focus is in Cluster small, which has the least number of the reference.

3.3 Data Migration

Once the banks are decided to shut down, dirty bit of each access in closed banks will be checked. If data in closed banks is dirty, that is to say, the data in DRAM cache is different from the data stored in main memory, the data in DRAM cache would write back to main memory. If data in closed banks is clean, drop it. Table 1 shows the bank configuration in DRAM cache.

Table 1. Processor and DRAM Cache configuration

Number of cores	1	L3 bus frequency	0.67 MHz
Frequency	2 GHz	L3 capacity	128 MB
L1 dcache/icache capacity	64 KB/32 KB	Bank	8 banks per rank
l2 Cache capacity	2 MB	Die	4
–	–	Row buffer size	2 KB

After we handle the data in the last two banks, if there is a memory access to these two banks, migrate it to the corresponding banks. In our work, the first three bits in access address index the bank ID. In migration process, for identifying the data which is originally access to the closed banks, we add a migration bit in DRAM cache. 0 is no migration, 1 represents migrated from other banks.

3.4 Remapping

When two banks are decided to be closed, the future access in bank has the smallest access number will remapping to the forth bank count backwards. The future access in bank has the last but one reference number will remapping to the third bank count backwards. What is more, before remapping begins, we set a migration bit in order to know whether the data is migrated or not. Upon an access comes, it maps to a bank. If this bank is open, migration bit stay the same. All the access behavior does not change as well. But when the access mapped to a closed bank. The migration bit switch to 1. In the meantime, tag bits stay still, only bank bits change.

3.5 Reopen

At every 10,000 ticks, we check the access pattern again. In the new 5000 ticks, if the distribution of clusters changes, that is, the Small cluster is no more the same, we will reopen the closed banks and do our power management mechanism over again.

3.6 Example

In application 464.h264ref. Its access pattern shows bank 2, 5, 6 and 7 are in Small cluster. Bank 1 and 3 are in Medium cluster. The rest bank 0 and bank 4 in Big cluster. Then we calculate and get that the reference number in Small cluster occupies 0.073 of Medium cluster. Thus, migrate the data in bank 7 (has least access number) to bank 2 (the forth bank count backwards according to reference number), bank 6 (the last but one) to bank 5(the third bank count backwards). Meanwhile, set the migration bit of those migrated data bit to 1. At last, data migration finished.

3.7 Hardware Implementation

Figure 5 shows the possible hardware implementation of our Coarse Granularity Data Migration power management mechanism. Every bank has a gating signal to control the

state of the bank (open or close). In each rank, there are four banks. A memory controller is used to manipulate which rank is activated. *Rank_Sel* chooses the determined rank. In order to select the desired bank, rank number and bank number are both needed to be confirmed. In other words, we use *Rankx_Bankx* to choose the bank we want.

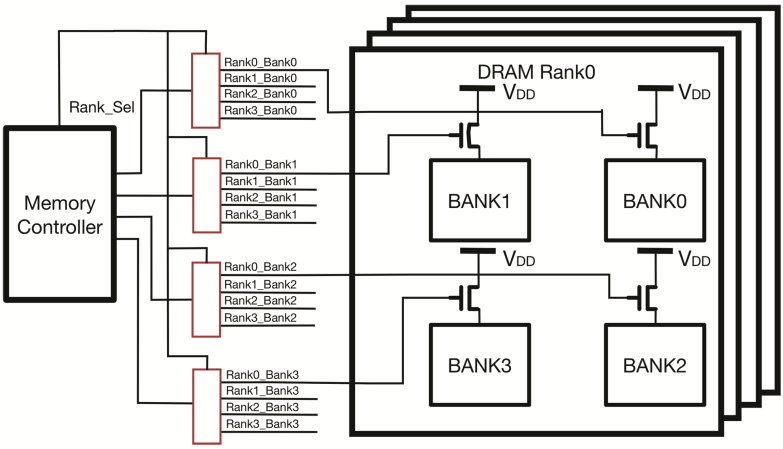

Fig. 5. Hardware implementation of power management

4 Experimental Evaluation

4.1 Experiment Setup

In our experiments, we use GEM5 simulator with a detailed memory model. The instruction set used is ARM V7 ISA. Table 1 gives the configuration used in our work. The 3D DRAM Cache used in the experiments is 128 MB.

Experiment Platform. The experiment platform is shown in Fig. 6. It contains 5 major parts: The processor simulator (Gem5), 3D DRAM Cache model (3D DRAM Cache),

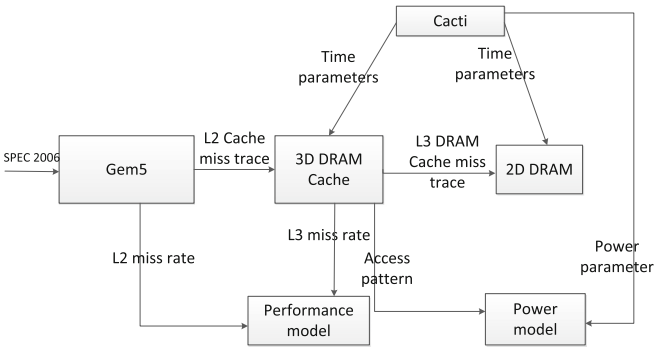

Fig. 6. Infrastructure of the whole experimental platform

main memory model (2D DRAM), power model (Power model), performance model (Performance model). Cacti offers timing and power parameter for our 5 parts except Gem5. Gem5 is configured to generate complete DRAM access traces (after L2 cache).

3D DRAM cache models: We exploit a DRAM simulator DRAMSim2 [6] to implement the trace-driven DRAM cache simulator: Thick Cache. This 3D-DRAM cache is an Alloy Cache [2] based Cache. The 3D DRAM Cache parameters are given by Cacti-3dd [7]. These parameters are listed in Table 2.

Table 2. CACTI-3DD configuration and parameter result

	2D main memory	3D DRAM Cache
Configuration		
Capacity	2G	128 MB
Frequency	677 MHz	677 MHz
Die	1	4
Parameter(latency)		
t_RCD	5.9164 ns	4.37031 ns
t_RAS	20.3714 ns	5.44791 ns
t_RC	34.5217 ns	6.68482
t_CAS	9.85235 ns	6.54791 ns
t_RP	14.1503 ns	1.7791 ns
Parameter(energy)		
Activate energy	0.489495 nJ	0.076035 nJ
Read/write energy	0.543979 nJ	0.367355 nJ
Prefetch energy	0.405572 nJ	0.065765 nJ

Main Memory model: The 2D main memory model is simulated by DRAMSim2 [6]. The configuration and timing parameters for DRAMSim2 are generated by Cacti [7].

Power model: The number of access of the 3D DRAM Cache during simulation is counted in the Thick Cache simulator. Then the following equation is used to calculate the power consumption:

$$\text{Dynamic power} = \text{read energy} * \text{read count} + \text{write energy} * \text{write counts} \quad (1)$$

$$\text{Leakage power} = \text{execution time} * \text{leakage energy} \quad (2)$$

Performance model: This model is used to calculate the average memory access time of the 3D DRAM Cache and the baseline system.

Benchmark. In this work, we use 164.gzip/175.vpr/181.mcf from SPEC CPU 2000 and 456.hmmer/458.sjeng/473.astar from SPEC CPU 2006 as our benchmark to estimate the performance and power of the proposed architecture. In each application, 2 million ticks are used as warmup time. 200 million instruction simulated in the whole process.

We get the L2 miss trace generated by Gem5. Then the traces are fed into our Thick Cache simulator to do the performance and power estimation.

4.2 Baseline System

The baseline system in this work is a processor core with L1 Cache and L2 Cache, without L3 Cache and a plane main memory based on DRAMSim2. The common parts of these two architectures share the same parameter of each hierarchy, except L3 Cache. The parameters of processor core and memory system are listed in Tables 1 and 2.

4.3 Implementation

For the one and only difference between baseline system and 3D-stacked system is a L3 3D DRAM Cache, a Cache simulator Dinero [8] is used to verify the accuracy of the DRAM Cache. We use the same configuration and traces to DRAM Cache and Dinero, miss rates are compared to see if our DRAM Cache is right. As is depicted in Fig. 7, we can see our Thick Cache simulator is basically accurate.

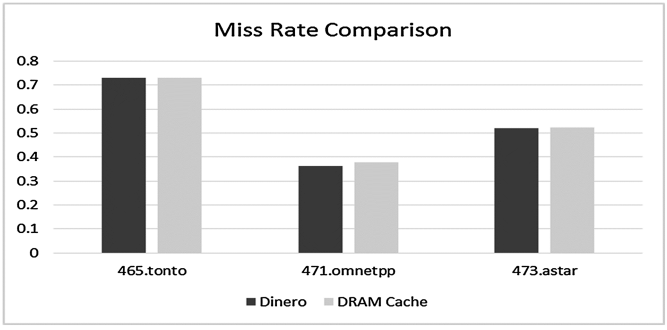

Fig. 7. Miss ratio comparison between Dinero and Thick Cache

4.4 Experimental Result

Performance Comparison. For our proposed 3D-DRAM LLC system, we got the LLC average access time and use it as the metric of performance. The results are shown in Fig. 8.

As it is depicted in Fig. 8, the performance behavior in these two systems are almost the same. In some cases, such as 437.lesli3d and 471.omnetpp, 3D DRAM Cache system shows shorter access time. And in a 3D DRAM Cache system, it has bigger capacity and no high area consumption, which makes our optimizing system a proper way to upgrade memory system.

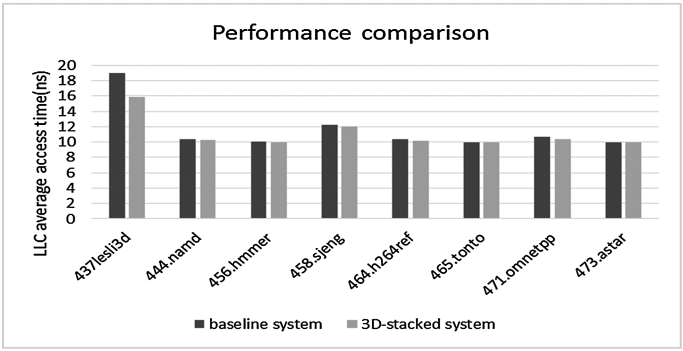

Fig. 8. Performance comparison between baseline and 3D DRAM Cache system

Power Comparison. In this section, we analysis the power consumption of the proposed design, including dynamic power and leakage power. Dynamic power results from writing and reading operations, which is determined by the total amount of the two kinds of operation. Leakage power comes from the leakage current of DRAM cells.

From the results of SPEC2006, it can be seen that for last level cache, the number of writing and reading operations are not very large, thus contributing to the fact that the dynamic power consumption of our design, which is shown in Fig. 9, is far less than its leakage power consumption, so we can take the leakage power consumption as the total power consumption approximately.

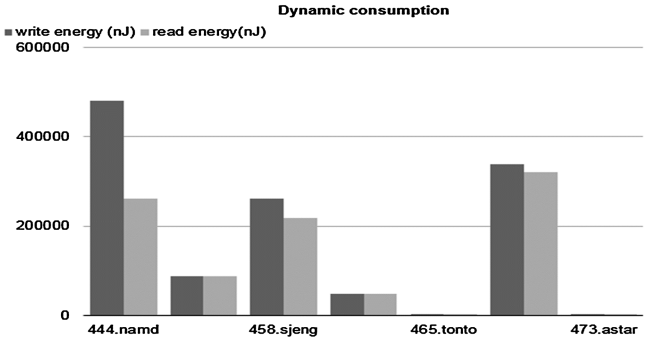

Fig. 9. Dynamic power consumption

In order to reduce its leakage power, we adopted bank migration and partial power down. The power results after optimization shown in Fig. 10 are much less. We can see that the leakage power consumption has decreased obviously, which prove that our solution is effective.

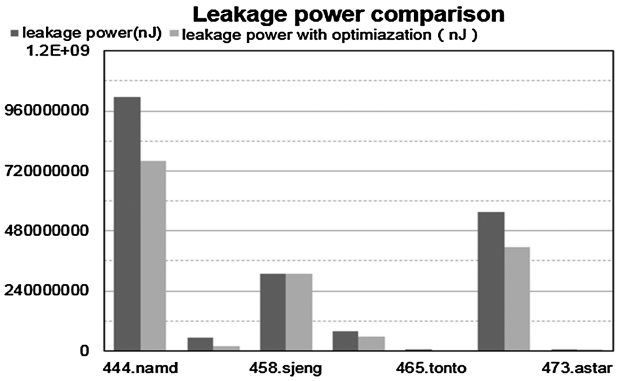

Fig. 10. Comparison between leakage power before power management and after

5 Related Work

5.1 DRAM Cache

In previous research, many studies have been explored the model of DRAM cache. In order to reduce the miss rate, Cached DRAM [22] integrates SRAM cache in the DRAM memory to exploit its locality in memory accesses and storage efficiency. In such DRAM caches, SRAM tags are placed in the stacked DRAM along with the data blocks [21–23]. However, this can potentially require two DRAM accesses per cache lookup (one for the tag look up and one for data). Thus, a state-of-art DRAM cache method, Alloy Cache [2] proposed to reduce latency. It organizes as a direct-mapped organization. Although reduce the hit rate, improving the cache access latency greatly. Our DRAM cache model is inspired by a latency-optimized cache architecture, named Alloy cache [2] which bursts tag and data in a single stream to wipe out the tag serialization delay.

5.2 Power Optimization of DRAM Cache

DRAM power management approaches basically developed into two parts: those aim to solve dynamic power such as memory traffic reshaping and increase the locality of memory reference [9], and those try to figure out leakage power by using decrease memory access or memory footprint [12]. Recently, there have been a large number of researches using adaptive power saving ability offered by several-banked DRAM [9–11].

Dynamic Power Management. For saving dynamic power, some techniques reduce the number of access to the specific memory level by using additional storage structures [13]. Some techniques utilize frequent accessed data with lower energy mode to reduce dynamic power per access [14]. Some other techniques perform tag bits match in several-step manner, while other techniques cut down the required bits for comparison [15]. Also, the techniques mentioned above can be extended to utilize in multiprocessor systems. In 3D multicore systems, Meng et al. [16] proposed a runtime optimization technique to maintain performance and power consumption in the same time.

Leakage Power Management. For saving leakage power, some researchers adaptive selecting a part of the cache to reduce the power. Based on the power-off granularity, leakage power management can be classified to way-level, set-level (or bank-level) [17], cache block-level [18], cache sub-block level [19] or cache sub-array level [12] etc. In 3D integration memory systems, in order to solve temperature and high power consumption, Woojin Yun and Jongpil Jung et al. [20] propose a dynamic voltage and frequency scaling (DVFS) scheme which can be adapted to cache bank or a group of cache banks for 3D-stacked L2 DRAM cache. Thus, they can obtain the supply voltages of different cache zones (or banks).

6 Conclusion

A key design methodology for improving the performance of processor core lies in breaking the Memory Wall. In our work, a L3 3D DRAM Cache is used in the new memory hierarchy to obtain higher capacity and wider bandwidth. However, for each 3D system, power consumption is a critical issue for a better performance. As for a big capacity LLC, leakage energy consists a big portion of the whole energy consumption. Thus, a bank closing mechanism is adopted to decrease leakage energy in our design. First of all, we analyze the access pattern offline of some applications from SPEC CPU 2006, determines which banks are not likely used in execution. Then, we shut them down. At last, static power consumption of some application decreased to 0.75x, a portion of application reach to 0.375x. In the future, in order to decrease the dynamic power in LLC, the migration of data in the process of closing banks can be put in to a further study.

References

1. Loh, G.H., Hill, M.D.: Efficiently enabling conventional block sizes for very large die-stacked dram caches. In: Proceedings of Annual International Symposium Microarchitecture, pp. 454–464 (2011)
2. Qureshi, M.K., Loh, G.H.: Fundamendal latency trade-off in architecture dram caches: out performing impractical sram-tags with a simple and practical design. In: 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture, pp. 235–246 (2012)
3. Jevdjic, D., Loh, G.H., kaynak, C., Falsa, B.: Unison cache: a scalable and effective die-stacked dram cache. In: 2014 47th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), pp. 25–37 (2015)
4. Su, C., Despain, A.: Cache design tradeoffs for power and performance optimization: a case study. In: Proceedings of International Symposium on Low Power Design, pp. 63–68 (1997)
5. Hasegawa, A., Kawasaki, I., Yamada, K., Yoshioka, S., Kawasaki, S., Biswas, P.: Sh3: high code density, low power. IEEE Micro **15**(6), 11–19 (1995)
6. Rosenfeld, P., Cooper-Balis, E., Jacob, B.: DRAMSim2: a cycle accurate memory system simulator. IEEE Comput. Archit. Lett. **10**(1), 16–19 (2011)
7. Chen, K., Li, S., Muralimanohar, N., et al.: CACTI-3DD: architecture-level modeling for 3D die-stacked DRAM main memory. In: Proceedings of the Conference on Design, Automation and Test in Europe, pp. 33–38. EDA Consortium (2012)
8. http://self.gutenberg.org/articles/dinero_(cache_simulator)

9. Amin, A., Chishti, Z.: Rank-aware cache replacement and write buffering to improve DRAM energy efficiency. In: Proceedings of the 16th ACM/IEEE International Symposium on Low Power Electronics and Design, pp. 383–388. ACM (2010)
10. Ware, M., Rajamani, K., Floyd, M., Brock, B., Rubio, J., Rawson, F., Carter, J.: Architecting for power management: the IBMR POWER7 approach. In: HPCA, pp. 1–11. IEEE (2010)
11. Chandrasekar, K., Akesson, B., Goossens, K.: Runtime power-down strategies for real-time SDRAM memory controllers. In: Proceedings of the 49th Annual DAC, pp. 988–993. ACM (2012)
12. Yang, L., Dick, R.P., Lekatsas, H., Chakradhar, S.: Online memory compression for embedded systems. ACM Trans. Embed. Comput. Syst. **9**, 1–30 (2010)
13. Kin, J., Gupta, M., Mangione-Smith, W.: The filter cache: an energy efficient memory structure. In: 30th International symposium on Microarchitecture (MICRO), pp. 184–193 (1997)
14. Udipi, A., Muralimanohar, N., Balasubramonian, R.: Non-uniform power access in large caches with low-swing wires. In: International Conference on High Performance Computing (HiPC). IEEE, pp. 59–68 (2009)
15. Kwak, J., Jeon, Y.: Compressed tag architecture for low-power embedded cache systems. J. Syst. Archit. **56**(9), 419–428 (2010)
16. Meng, J., Kawakami, K., Coskun, A.K.: Optimizing energy efficiency of 3-D multicore systems with stacked DRAM under power and thermal constraints. In: Design Automation Conference, pp. 648–655. ACM (2012)
17. Ku, J., Ozdemir, S., Memik, G., Ismail, Y.: Thermal management of on-chip caches through power density minimization. In: International Symposium on Microarchitecture (MICRO), pp. 283–293 (2005)
18. Kaxiras, S., Hu, Z., Martonosi, M.: Cache decay: exploiting generational behavior to reduce cache leakage power. In: 28th International Symposium on Computer Architecture (ISCA), pp. 240–251 (2001)
19. Alves, M.A.Z., et al.: Energy savings via dead sub-block prediction. In: International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD) (2012)
20. Yun, W., Jung, J., Kang, K., et al.: Temperature-aware energy minimization of 3D-stacked L2 DRAM cache through DVFS. In: Soc Design Conference, pp. 475–478 (2012)
21. Loh, G.H.: Extending the effectiveness of 3d-stacked dram caches with an adaptive multi-queue policy. In: Proceedings of the 42nd International Symposium on Microarchitecture, December 2009
22. Loh, G.H., Hill, M.D.: Efficiently enabling conventional block sizes for very large die-stacked dram caches. In: Proceedings of the 44th International Symposium on Microarchitecture, December 2011
23. Qureshi, M., Loh, G.H.: Fundamental latency trade-offs in architecting DRAM caches. In: Proceedings of the 45th International Symposium on Microarchitecture, December 2012

http://www.springer.com/978-981-10-2208-1

Advanced Computer Architecture
11th Conference, ACA 2016, Weihai, China, August 22-23, 2016, Proceedings
Wu, J.; Li, L. (Eds.)
2016, XII, 211 p. 121 illus., Softcover
ISBN: 978-981-10-2208-1