IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 1, 2018, accepted November 21, 2018, date of publication December 5, 2018, date of current version December 31, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2885064

Impacts of Memory Address Mapping Scheme on Reducing DRAM Self-Refresh Power for Mobile Computing Devices

ZONGWEI ZHU[1], JING CAO[1], XI LI[1], JUNNENG ZHANG[2], YOUQING XU[1], AND GANGYONG JIA[3]

[1]Suzhou Institute for Advanced Study, University of Science and Technology of China, Suzhou 215000, China
[2]School of Computer Engineering, Qingdao University of Technology, Qingdao 266000, China
[3]Department of Computer Science, Hangzhou Dianzi University, Hangzhou 310000, China

Corresponding authors: Zongwei Zhu (zzw1988@ustc.edu.cn), Jing Cao (congjia@mail.ustc.edu.cn), and Xi Li (llxx@ustc.edu.cn)

ABSTRACT With the growth of the Internet of Things (IoT), increasingly, more computing tasks are implemented on power-sensitive mobile devices, causing a bottleneck on energy consumption. Most mobile devices consume considerable power in the standby mode, during which the capacitive DRAM cells' self-refresh power, which is used to preserve data integrity, accounts for a large part. To address this issue, strategies from both hardware and software perspectives have been proposed, and yet, the existing hardware methods usually have a high cost. Software strategies mainly focus on the operating system page allocation strategy to maximize resource idleness. The partial array self-refresh technique has been proposed to allow for some of the DRAM cells to retain the self-refresh state while shutting down the others. Using this technique, numerous studies extend the unused DRAM idle time by clustering the applications' data but do not discuss the impact of the address mapping mechanism on power consumption. However, this impact was proven to be essential in our previous study. In this paper, we attempt to investigate the impacts of different memory mapping schemes on different clustering strategies using a hierarchy-based memory management system (HBMM) and provide insights for selecting the optimal schemes. The experimental results on the Android platform demonstrate that the schemes and strategies are closely correlated, and memory idle time can be prolonged 125 times with their suitable combination. Conversely, the worst scheme may decrease memory idleness by 12% compared with the original system. Furthermore, our studies on a physical platform show that standby power consumption can be saved by 23% when using HBMM combined with the best clustering strategy.

INDEX TERMS DRAM, mapping scheme, operating system, self-refresh power.

I. INTRODUCTION

With the development of IoT, mobile devices have morphed into general-purpose computing platforms; as a result, energy has become a first-class design constraint [1]–[4]. Saving energy is one of the best ways to enhance the mobility of mobile computing devices.

The liquid crystal display (LCD), backlight and CPU are typically the highest consumers of the mobile devices' total power in active mode. However, from the perspective of custom groove-in, mobile computing devices such as monitors and smartphones are idle (standby and sleep) most of the time [15], and the highest power consumers mentioned above should be shut down. In the meantime, with the DRAM cell being periodically recharged, the self-refresh power naturally becomes the dominant consumer. A study in 2003 illustrates that power consumed by DRAM accounts for 35% of the power consumption in smart devices [5]. Further, the heavy implementation of computing-intensive applications such as video-based and AI-based ones have caused increasing demand for producing memory with larger capacity and larger bandwidth. All these demands are making the energy-efficiency problem much more prominent. For example, the bandwidth of LPDDR4 used on Apple's A9x and A10 platform has reached 51.2 GB/s [6]; the Monolith Chaconne smartphone to be published in 2018 will deploy 18 GB memory [7]. In the past five years, there have been many papers concerning this problem at top international conferences such as HPCA, ASPLOS, ISCA, DATE, and ISLPED,

2169-3536 © 2018 IEEE. Translations and content mining are permitted for academic research only.
Personal use is also permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

meaning that it has apparently become a hot topic in both industry and academia.

There are plenty of previous studies that have proposed to reduce the refresh power from various perspectives.

In the hardware field, for example, Flikker [8] reduced hardware reliability by applying different refreshing strategies to critical and noncritical data to save power. A modification to the DRAM proposed by Chang *et al.* [10] added the DRAM's internal refresh counter into its existing control-register access protocol. There are also some studies on optimizing the memory controller. Ghosh and Lee [11] introduced a timeout counter for each row in memory. Ashish Ranjan et al. increased the number of free memory areas by modifying the memory controller to provide support for data compression and decompression [12]. However, these techniques require hardware support, which is costly or even infeasible for battery-based mobile devices.

For software methods, Sudharsan et al. reduced the number of wake-up banks during half-write by modifying the memory mapping mechanism [13]. Our previous work studied the optimization of the page allocation strategy at the operating system (OS) level [14], which mainly focused on the active mode. The software optimization methods under active mode have guiding significance for power optimization in standby mode. Some application programming models, such as the Android platform, emphasize reducing DRAM usage in idle mode [16], which focuses on maximizing DRAM idleness based on the page allocation mechanism. Further, some manufacturers have produced the new generation of low-power DRAM (LPDDR), and a new feature called partial array self-refresh (PASR) is developed. PASR enables the DRAM to retain state in only part of the memory, thus further reducing the self-refresh power. There are some classic PASR-based solutions, such as the single-/dual-ended bank PASR and the bank-selective PASR [17]. Because the bank-selective PASR covers smaller granularity and each bank can be independently marked to be self-refreshed, it has become the most attractive methodology and has been adopted by Hynix, Samsung, Elpida and Micron DRAM manufacturers. However, although many studies aggregate data from the perspective of the OS, they are often based on the assumption of a linear address mapping mechanism and do not discuss the impact of the address mapping mechanism on power consumption. Our investigation results illustrate that the impact of the address mapping mechanism on PASR is still noticeably large and has not been well resolved by previous solutions.

In summary, previous energy-optimizing methods can be divided into hardware and software categories. The hardware optimization method is usually costly. At the operating system (OS) level, to coordinate with PASR more effectively, most similar works mainly focus on prolonging memory idle time by clustering data together via proper data allocation or dynamic data migration [18]–[20]. However, there are no studies considering the impacts of memory mapping schemes on data clustering strategies, which is significant in designing the power-efficient system. To build power-efficient systems, this study focuses on reducing the refresh power using the bank selective PASR strategy for mobile devices. Our experimental results show that the optimal scheme-strategy matching can increase memory idle time 125 times, while the worst strategy may decrease by 12%. Furthermore, to confirm HBMM's effectiveness, numerous experiments are also performed on a real platform, and the results show that it can save 23% standby power consumption with the best clustering strategy.

The rest of this paper is organized as follows. After discussing the proposed mechanisms in detail in Section II, we present our methods of experimental studies and analyse the results in Section III. Section IV summarizes the related work, and finally, Section V concludes the paper.

II. PROPOSED MECHANISMS

A bank is the basic unit of PASR, and in the DRAM memory address translation procedure, the address mapping mechanism determines which bank to place pages into.

A. HIERARCHY-BASED MEMORY MANAGEMENT i SYSTEM (HBMM)

The buddy system is a classic memory resource manager having excellent allocation efficiency. In the buddy system, continuous 2^{order} free pages (called a page block) are organized in the *free_area* list with the corresponding order, which ranges from 0 to a specific upper limit.

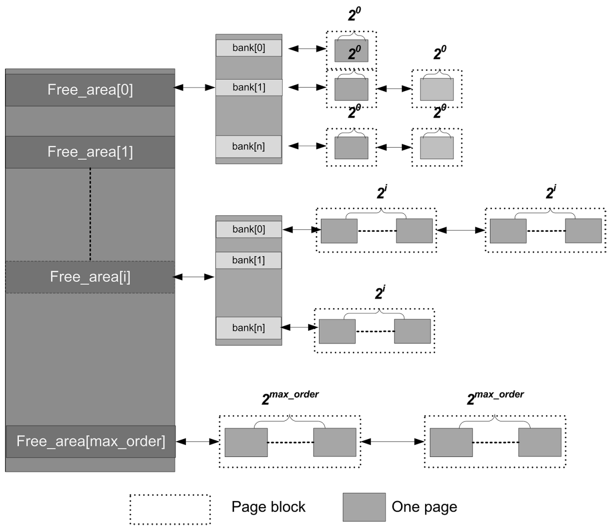


FIGURE 1. Bank-sensitive buddy system structure.

When threads request a size of 2^{order} pages, the algorithm will return the selected pages if they are found using the *find_page* function from the *free_area[order]*; otherwise, $2^{order+1}$ pages will be removed from the *free_area[order+1]* and be divided into two equal page blocks, where one is allocated, and another is inserted into the *free_area[order]*. Originally, the buddy system treats all of the page blocks in one *free_area* list equally, and the physical memory is a black box. As is shown in Fig. 1, a hierarchy-based memory management system (HBMM) is introduced, and we modify the original free list organization into a hierarchy: for each order of the *free_area* list, we reorganize the free page blocks to form *Bank[0..n]* lists according to its bank identification. The bank table *sys_bank_array[0..number_banks]* is maintained to support different clustering strategies.

In light of the previous work in [18]–[20], four strategies cooperated with different *sys_bank_array* organizations and are summarized as follows:

- *Utilization strategy*: To create more bank idleness, the table will be sorted by the utilization of the banks in descending order.
- *LRU strategy*: To fully exploit memory access locality, the bank table is organized according to the basis of the bank access time.
- *Ascend strategy*: The bank table is sorted according to the bank identification in ascending order, which fills an entire bank before moving on to the next.
- *Descend strategy*: The bank table is sorted in descending order in contrast to the ascend strategy.

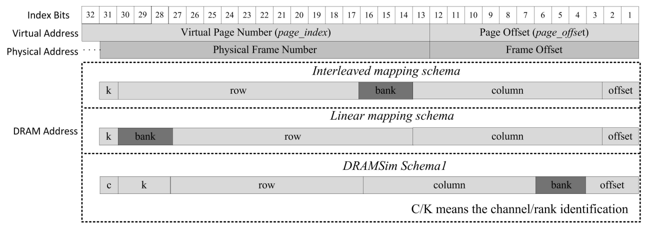
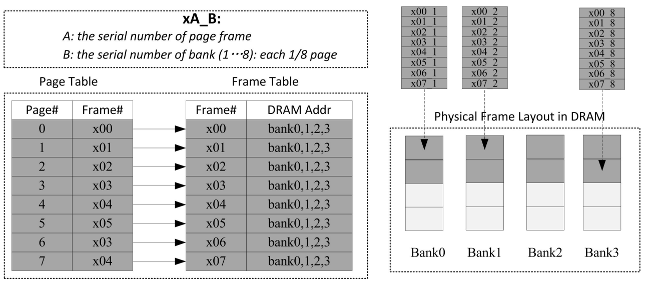


FIGURE 2. Different classic mapping schemes.

B. HBMM COMBINED WITH DIFFERENT MAPPING SCHEMES

An address-translating mechanism establishes the connections between the physical and the DRAM address, and the mapping scheme directly determines which bank to place pages into. Fig. 2 gives the correspondence of the virtual address, physical address and DRAM address. The virtual address and the physical address have fixed formats. The DRAM address has different formats according to the mapping schemes. In the DRAM address row (a group of storage cells activated in parallel in response to a row activation command), it shows the interleaved and linear mapping scheme used by the *HiSilicon K3V2* product [33] along with the default scheme adopted by *DRAMSim* [21], which is an open-source joint-electron device engineering council (JEDEC) DDR memory system simulator. The memory architecture can be divided into a low-bit multi-access cross-memory (LMCM) architecture and a high-bit multi-access cross-memory (HMCM) architecture according to the position of bank bits in virtual address [14].

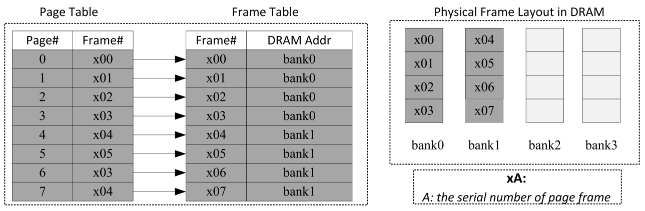
Based on the above classification, *DRAMSim's* default scheme belongs to LMCM. Fig. 3 describes an example of LMCM whose granularity is *page_size*/8 = 512 Bytes (the default page size is 4 KB in Linux, while some platforms support big page mode whose size is 4 MB). However, since bank bits fall in the range of the *page_offset* on LMCM,


xA_B:
A: the serial number of page frame
B: the serial number of bank (1 ⋯8): each 1/8 page
Page Table
Frame Table
x00_1 x01_1 x02_1 x03_1 x04_1 x05_1 x06_1 x07_1
x00_2 x01_2 x02_2 x03_2 x04_2 x05_2 x06_2 x07_2
x00_8 x01_8 x02_8 x03_8 x04_8 x05_8 x06_8 x07_8
Physical Frame Layout in DRAM
Bank0
Bank1
Bank2
Bank3

Page#	Frame#
0	x00
1	x01
2	x02
3	x03
4	x04
5	x05
6	x03
7	x04

Frame#	DRAM Addr
x00	bank0,1,2,3
x01	bank0,1,2,3
x02	bank0,1,2,3
x03	bank0,1,2,3
x04	bank0,1,2,3
x05	bank0,1,2,3
x06	bank0,1,2,3
x07	bank0,1,2,3

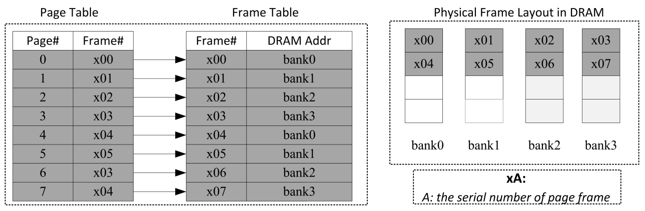
FIGURE 3. 512-Byte LMCM architecture.



Page#	Frame#
0	x00
1	x01
2	x02
3	x03
4	x04
5	x05
6	x03
7	x04

Frame#	DRAM Addr
x00	bank0
x01	bank0
x02	bank0
x03	bank0
x04	bank1
x05	bank1
x06	bank1
x07	bank1

FIGURE 4. Linear mapping scheme.



Page#	Frame#
0	x00
1	x01
2	x02
3	x03
4	x04
5	x05
6	x03
7	x04

Frame#	DRAM Addr
x00	bank0
x01	bank1
x02	bank2
x03	bank3
x04	bank0
x05	bank1
x06	bank2
x07	bank3

FIGURE 5. Interleaved mapping scheme.

the size of the basic management unit is less than the page size. In other words, idle banks may rarely appear, and PASR would become inapplicable for optimizing the power regardless as to whether it is integrated with page migration or allocator mechanisms [18]–[20]. To address this issue, hardware modification is required, and we proposed the solution in our previous work [14]; it is beyond the scope of this paper.

Compared with the *DRAMSim's* default scheme, the linear and interleaved mapping scheme will be classified as HMCM. In the linear scheme, the physical address will be mapped to the DRAM address in the order of bank, row, column and width, and an application's data would typically concentrate on as few banks as possible as Fig. 4 shows. The interleaved mapping scheme maps the physical address in the order of row, bank, column and width. As illustrated in Fig. 5, a thread will generate multiple memory requests to different memory banks, which hide the latencies by overlapping accesses to many banks and then dramatically increases performance but consumes more power. As a result, an interleaved scheme could not perform the same clustering result as a linear scheme. Hence, clustering strategies play a much more important role and should be carefully selected compared with the linear one.

III. EXPERIMENTAL RESULTS

The evaluation is based on the VMware [22] simulator, and the operating system is Androidx86 4.1. The hardware configuration includes a quad-core CPU and 2 GB DRAM integrated with 16 banks. To evaluate the impacts of the mapping scheme on different HBMM strategies' effectiveness, three test scenarios are supplied. The first scenario corresponds to the OS startup process, the second test randomly requests 512 MB memory, and the last test is a 1-GB random memory allocation. The memory architecture of Fig. 2 is considered in the experiments: the linear mapping scheme uses the physical address' 28th-31st bits as the bank identification, and the interleaved scheme applies the 14-16th bits and the 31st bit to identify the bank number.

A. PERFORMANCE LOSS

HBMM will update the *sys_bank_array* to track all the banks' information, e.g., utilization and accessed state, which directly leads to the *O(n)* time complexity where n is the number of banks. To stressfully evaluate the performance impacts, *Ring dream factory, Sogou input method, Shredder chess, Youdao dictionary, Moboplayer video player, IQIYI video and Backgammon game*, these seven popular applications are selected. Fig. 6 depicts the absolute difference of the different applications' performance compared with the original system. The results prove that our algorithm's performance loss is within 4.5%.

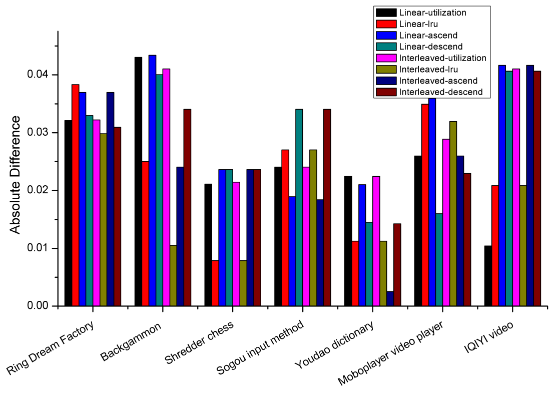


FIGURE 6. **Difference of application's performance compared with the original system.**

B. BANK IDLE TIME EVALUATION

To reduce power consumption, one common idea is to use as few resources as possible. In other words, memory management should cluster pages into fewer banks to maintain idleness as long as possible. Thus, the bank idle time can be an important factor to evaluate the HBMM's clustering effect. It is possible to calculate the refresh power consumption through the bank idle time according to the micron supported DRAM power calculation method [35]. In this subsection, we will discuss the clustering effects of the different HBMM strategies on different mapping schemes.

- *Linear Mapping Scheme*: Fig. 7 (the x-axis shows the different HBMM strategies cooperating with the three test scenarios, and the y-axis represents the bank idle time ratio) depicts the bank idle time ratio cooperating with the HBMM's four *sys_bank_array* organizations

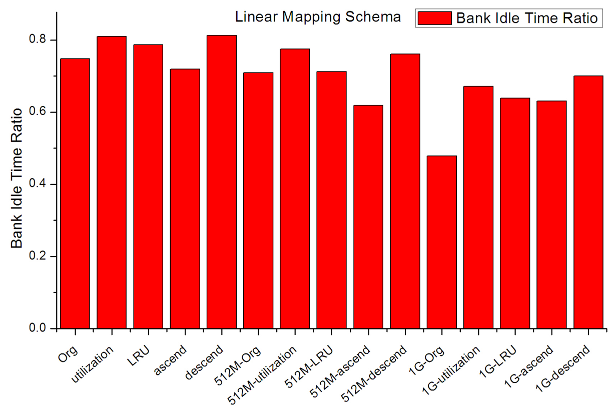


FIGURE 7. **Bank idle time ration in the linear mapping scheme.**

in three test scenarios. It is apparent that *utilization, LRU and descend strategies* have a weak effect in different scenarios. Even worse, the ascend strategy plays a negative role except in the 1-GB allocation scenario. The reason for this result is that when the mapping scheme is configured as a linear mapping scheme, the physical addresses will sequentially correspond with the DRAM address, and then, the sequent virtual address will also be mapped to the sequent banks to a great extent. Given that the kernel image will always be at the start of the DRAM, the Linux kernel maximizes the contiguous space by allocating runtime memory from the end of the physical DRAM moving downward. During the system boot-up stage, there may be so few running processes that the utilization and LRU strategies will cluster pages into some specific banks similar to the original policy. However, the ascend strategy is contrary to the original policy and allocates pages from bottom to top DRAM addresses; as a result, memory accesses will spread into more banks and cannot achieve the intended clustering effect. It should be pointed out that when the allocation reaches 1 G in size, more processes are created or freed, and the default Linux sequential principle will be broken, but all of the clustering strategies can hold their clustering effect and directly talk to DRAM, and then, more idleness will be created compared with the original Linux system.

- *Interleaved Mapping Scheme*: Fig. 8 (the x-axis shows the different HBMM strategies cooperating with the three test scenarios, and the y-axis represents the bank idle time ratio) shows the three test scenarios' bank idle time ratio of the four HBMM strategies on the interleaved mapping scheme. It can be concluded that the utilization and ascend strategies are more efficient than LRU and the descend strategies in all tested scenarios. After the system starts up, the OS will occupy hundreds of megabytes of the memory region; as a result, bank 0-7 will be pre-allocated to the OS due to interleaved mapping scheme. The results of Fig. 9 (the x-axis shows the different HBMM strategies cooperating with

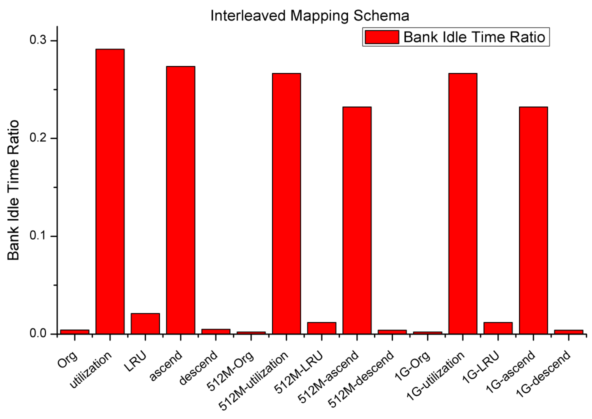


FIGURE 8. Bank idle time ration in linear mapping scheme.

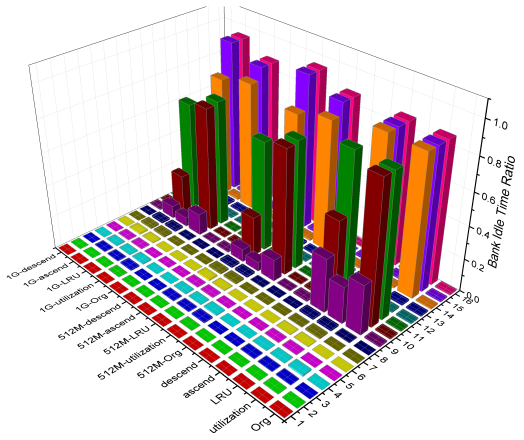


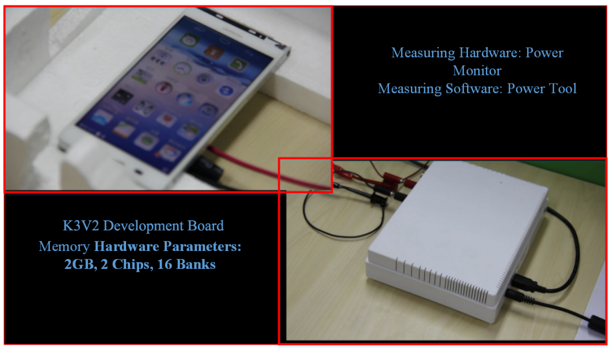
FIGURE 9. Each bank idle time ratio on interleaved mapping scheme.

the three test scenarios, and the y-axis represents each bank number, and the z-axis gives the bank's idle time ratio) demonstrate that the descend strategy also prefers to utilize the former banks, and then, it is difficult to gain profit. Though applications may perform a good localization on their virtual address space, the interleaved scheme will result in different physical banks being switched in turn to achieve contiguous virtual address accesses. In summary, the LRU strategy updates the *sys_bank_array* table as soon as possible and performs poorly in regard to the clustering result, thus gaining non-obvious energy efficiency.

In summary, the experimental results have demonstrated that the memory mapping scheme plays an important role in HBMM, and the worst selected strategy may even decrease the bank idle time by 12%. The Linux kernel has already performed better on the clustering effect on the linear mapping scheme while performing poorly on the interleaved scheme. It is worth mentioning that the utilization strategy may be the best candidate for both schemes, especially for the interleaved one.

C. HBMM'S EFFECTIVENESS ON A REAL PLATFORM

To evaluate HBMM's maximum effectiveness, we adapt the best suitable utilization strategy. Fig. 10 illustrates HBMM's



Measuring Hardware: Power Monitor
Measuring Software: Power Tool
K3V2 Development Board
Memory Hardware Parameters:
2GB, 2 Chips, 16 Banks

FIGURE 10. Real hardware platform and measuring system environment. The left part is a mobile phone, and the right part is the power measuring hardware and software.

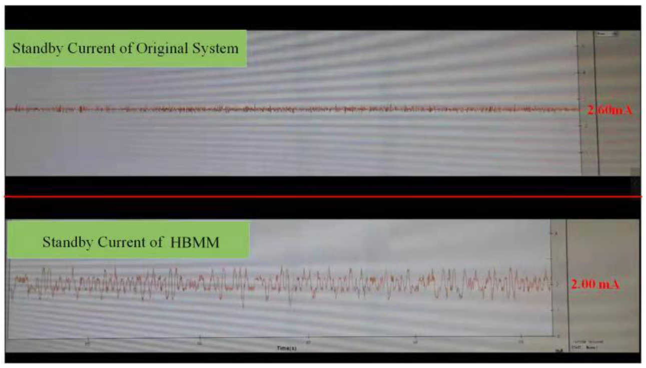


FIGURE 11. The original system and HBMM's standby current.

real test platform and its corresponding power measurement system in detail. The *K3 V2* development board, as the experimental testbed, supports the interleaved scheme by default, and its mapping detail is shown in Fig. 2. The Monsoon Solution power measurement system [36] including hardware and software tools is used to measure the system standby power. To avoid other interferences, the system's standby current will be plotted after the system enters the standby state for 5 minutes.

As power consumption equals the product of system voltage and current, and the Monsoon system is in charge of supplying constant system voltage, typically 5.0 v, the system standby power will synchronize with the current. It can be concluded from Fig. 11 that HBMM's standby current is approximately 2.00 mA, while that of the original system is approximately 2.60 mA. The saved power percent can be up to 23%.

There is one special case in which the system has insufficient idle memory, and HBMM may have difficulty achieving good performance. However, some mobile devices' OSs, such as Android, emphasize reducing application DRAM usage in idle mode. In other words, memory resources will be recycled to create more memory idleness, and HBMM can be easily integrated with other power-saving software.

IV. RELATED WORK

Mobile device hardware design techniques traditionally overprovision for the system's worst-case behaviour during computing tasks. However, the worst case behaviour is rarely exhibited in the majority of common usage scenarios, and therefore, a new class of techniques called the better-than-worst case (BTWC) [21] have emerged that provision for the average case and treat the worst case behaviour as an exception.

Razor [24], one of the best known BTWC examples, lowers the processor's voltage to such a point that the processor starts to experience errors due to timing violations. RAPID [25] and ESKIMO [9] are hardware-software techniques that apply the BTWC principle for DRAM refresh-power reduction. RAPID focuses on characterizing each physical page's leakage behaviour and divides the pages into different classes. ESKIMO saves DRAM power with the help of the knowledge of application semantics. Flikker [8] is different than RAPID and ESKIMO; it requires the programmers to specify critical and non-critical data in programs to reduce unnecessary refresh regions. Finally, similar to RAPID, ESKIMO and Flikker, many other techniques modify the memory controller hardware and memory allocator to expose the details of the application's allocation patterns for reducing redundant refreshes [11], [26], [27]. Meanwhile, numerous software-based techniques have been proposed to make a tradeoff between hardware reliability and power consumption in an application specific manner [28]–[31]. For example, Fluid-NMR [31] performs N-way replication (N is varied based on the ability to tolerate errors) of applications for tolerating errors due to reductions in the processor's voltage levels. Relax [29] is a technique to save computational power by exposing hardware errors to the software in specified regions of code; it allows programmers to mark certain regions of the code as relaxed and adjusts the processor's voltage and frequency when executing such regions. IshwarBhati et al. proposed a modification to the DRAM that extends its existing control-register access protocol to include the DRAM's internal refresh counter [10]. Byoungchan Oh et al. provided a solution to optimize the leakage current of DRAM cells by selectively applying different voltage levels to the DRAM cell transistors when they are active (accessed for refreshing) and idle (pre-charged) by adjusting both the word-line and body voltages [4]. However, the aforementioned techniques either require substantial changes to the memory controller's hardware or are not applicable to battery-based mobile devices.

To the best of our knowledge, the memory power mode control scheme and PASR technology are the most practical solutions that have been proposed by numerous manufacturers. The single-/dual-ended bank PASR and the bank-selective PASR focus mainly on refreshing only part of the memory, which is critical and allows the system to lose the noncritical data [17]. Some works at the OS level have also been proposed to prolong memory module idle time. They mainly focus on clustering data into fewer modules by proper data allocation or dynamic data migration. Lebeck *et al.* [20] were the first to propose two power-aware page allocation policies to increase the chances that the DRAM chips could be put into low power modes. Kruijf *et al.* [29] proposed migrating hot micro-pages using DRAM copy to co-locate frequently accessed micro-pages in the same row-buffer, but identifying and migrating the hot data are time-consuming and cause considerable performance loss.

In summary, to manage memory resources effectively, there exists a challenge in how to allocate pages in a specific memory region under the BTWC principle when building the power-aware memory management system. We believe that the impacts of the memory mapping scheme should be considered. Hence, our work could play a positive role in further works on this topic.

V. CONCLUSIONS

In this paper, our goal is to prolong the battery lifetime of mobile devices by decreasing the DRAM self-refresh power in standby mode. We first combine a hierarchy-based memory system (HBMM) with four previously different policies to cluster pages together to prolong the DRAM bank idle time. Then, by introducing the memory mapping scheme factor on these policies, we find that memory mapping schemes play an important role in saving DRAM self-refresh power. The original Linux kernel has already demonstrated an improved clustering effect for the linear mapping scheme while performing poorly on the interleaved scheme. Furthermore, experiments show that the utilization strategy may be the best candidate for both schemes and especially for the interleaved scheme. Our proposed strategy can also be applied to memory active mode and storage power control designs.

REFERENCES

[1] K. W. Cameron, R. Ge, and X. Feng, "High-performance, power-aware distributed computing for scientific applications," *Computer*, vol. 38, no. 11, pp. 40–47, Nov. 2005.

[2] C. Ellis, A. Lebeck, and A. Vahdat, "System support for energy management in mo-bile and embedded workloads white paper," Dept. Comput. Sci., Duke Univ., Durham, NC, USA, Tech. Rep., 1999.

[3] C. Lefurgy, K. Rajamani, F. Rawson, W. Felter, M. Kistler, and T. W. Keller, "Energy management for commercial servers," *IEEE Comput.*, vol. 36, no. 12, pp. 39–48, Dec. 2003.

[4] B. Oh, N. Abeyratne, J. Ahn, R. G. Dreslinski, and T. Mudge, "Enhancing dram self-refresh for idle power reduction," in *Proc. Int. Symp. Low Power Electron. Design*, 2016, pp. 254–259.

[5] M. A. Viredaz and D. A. Wallach, "Power evaluation of a handheld computer," *IEEE Micro*, vol. 23, no. 1, pp. 66–74, Jan./Feb. 2003.

[6] *Apple A10 Memory*. Accessed: Oct. 2018. [Online]. Available: https://soc.specshero.com/en/apple-a10-memory/model-2235-5

[7] *Turing Monolith Chaconne*. Accessed: Oct. 2018. [Online]. Available: https://www.gadgetsnow.com/mobile-phones/Turing-Monolith-Chaconne

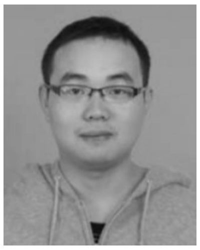
[8] S. Liu, K. Pattabiraman, T. Moscibroda, and B. Zorn, "Flikker: Saving DRAM refresh-power through critical data partitioning," *ACM SIGPLAN Notices*, vol. 47, no. 4, pp. 213–224, 2012.

[9] C. Isen and L. John, "Eskimo—Energy savings using semantic knowledge of inconsequential memory occupancy for DRAM subsystem," in *Proc. 42nd Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Dec. 2009, pp. 337–346.

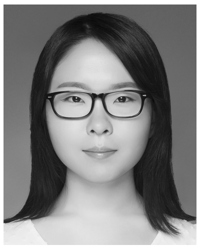
[10] K. K. Chang *et al.*, "Understanding reduced-voltage operation in modern DRAM devices: Experimental characterization, analysis, and mechanisms," in *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 1, no. 1, 2017, p. 10.

[11] M. Ghosh and S. HH Lee, "Smart refresh: An enhanced memory controller design for reducing energy in conventional and 3D die-stacked DRAMs," in *Proc. 40th Annu. IEEE/ACM Int. Symp. Microarchitecture*, Dec. 2007, pp. 134–145.

[12] A. Ranjan, A. Raha, V. Raghunathan, and A. Raghunathan, "Approximate memory compression for energy-efficiency," in *Proc. IEEE/ACM Int. Symp. Low Power Electron. Design (ISLPED)*, Jul. 2017, pp. 1–6.
[13] S. Jagathrakshakan, V. K. Tavva, and M. Mutyam, "Data remapping for an energy efficient burst chop in DRAM memory systems," in *Proc. 23rd Int. Conf. Parallel Archit. Compilation*, 2014, pp. 507–508.
[14] Z. Zhu, X. Li, C. Wang, and X. Zhou, "Memory power optimisation on low-bit multi-access cross memory address mapping schema," *Int. J. Embedded Syst.*, vol. 6, nos. 2–3, pp. 240–249, 2014.
[15] A. K. Karlson, B. R. Meyers, A. Jacobs, P. Johns, and S. K. Kane, "Working overtime: Patterns of smartphone and PC usage in the day of an information worker," in *Proc. 7th Int. Conf. Pervasive Comput.*, May 2009, pp. 398–405.
[16] M. Murphy, *Beginning Android*. New York, NY, USA: Apress, 2009.
[17] T. Brandt, T. Morris, and K. Darroudi, "Analysis of the PASR standard and its usability in handheld operating systems such as Linux," Intel, Santa Clara, CA, USA, Tech. Rep., 2007.
[18] V. De La Luz, M. Kandemir, and I. Kolcu, "Automatic data migration for reducing energy consumption in multi-bank memory systems," in *Proc. Design Autom. Conf.*, 2002, pp. 213–218.
[19] K. Sudan, N. Chatterjee, D. Nellans, M. Awasthi, R. Balasubramonian, and A. Davis, "Micro-pages: Increasing DRAM efficiency with locality-aware data placement," *ACM SIGPLAN Notices*, vol. 45, no. 3, pp. 219–230, 2010.
[20] A. R. Lebeck, X. Fan, H. Zeng, and C. Ellis, "Power aware page allocation," in *Proc. 9th Int. Conf. Archit. Support Program. Lang. Oper. Syst.*, 2000, pp. 105–116.
[21] D. Wang, B. Ganesh, N. Tuaycharoen, K. Baynes, A. Jaleel, and B. Jacob, "DRAMsim: A memory system simulator," *ACM SIGARCH Comput. Archit. News*, vol. 33, no. 4, pp. 100–107, 2005.
[22] M. Rosenblum, "VMware's virtual PlatformTM," in *Proc. Hot Chips*, 1999, pp. 185–196.
[23] T. Austin, V. Bertacco, D. Blaauw, and T. Mudge, "Opportunities and challenges for better than worst-case design," in *Proc. Asia South Pacific Design Autom. Conf.*, 2005, pp. 2–7.
[24] D. Ernst *et al.*, "Razor: A low-power pipeline based on circuit-level timing speculation," in *Proc. 36th Annu. IEEE/ACM Int. Symp. Microarchitecture*, Dec. 2003, pp. 7–18.
[25] R. K. Venkatesan, S. Herr, and E. Rotenberg, "Retention-aware placement in DRAM (RAPID): Software methods for quasi-non-volatile DRAM," in *Proc. Int. Symp. High-Perform. Comput. Archit. (HPCA)*, 2006, pp. 155–165.
[26] J. Kim and M. C. Papaefthymiou, "Block-based multiperiod dynamic memory design for low data-retention power," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 11, no. 6, pp. 1006–1018, Dec. 2003.
[27] K. Patel, E. Macii, M. Poncino, and L. Benini, "Energy-efficient value-based selective refresh for embedded DRAMs," in *Proc. 15th Int. Conf. Integr. Circuit Syst. Design., Power Timing Modeling Optim. Simulation (PATMOS)*, 2005, pp. 466–476.
[28] W. Baek and T. M. Chilimbi, "Green: A framework for supporting energy-conscious programming using controlled approximation," in *Proc. ACM SIGPLAN Conf. Program. Lang. Design Implement. (PLDI)*, 2010, pp. 198–209.
[29] M. D. Kruijf, S. Nomura, and K. Sankaralingam, "Relax: An architectural framework for software recovery of hardware faults," in *Proc. 37th Annu. Int. Symp. Comput. Archit. (ISCA)*, 2010, pp. 497–508.
[30] H. Hoffmann, S. Misailovic, S. Sidiroglou, A. Agarwal, and M. Rinard, "Using code perforation to improve performance, reduce energy consumption, and respond to failures," Dept. Elect. Eng. Comput. Sci., Massachusetts Inst. Technol., Cambridge, MA, USA, Tech. Rep., 2009.
[31] J. Satori, J. Sloan, and R. Kumar, "Fluid NMR—Performing power/reliability tradeoffs for applications with error tolerance," in *Proc. Workshop Power Aware Comput. Syst. (HotPower)*, 2009, pp. 1–5.
[32] S. P. Muralidhara, L. Subramanian, O. Mutlu, M. Kandemir, and T. Moscibroda, "Reducing memory interference in multicore systems via application-aware memory channel partitioning," in *Proc. 44th Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Dec. 2011, pp. 374–385.
[33] *The Datasheet of HiSilicon K3V2 Chip*, Huawei Technologies Co. Ltd, Shenzhen, China, 2014.
[34] X. Fan, C. Ellis, and A. Lebeck, "Memory controller policies for DRAM power management," in *Proc. Int. Symp. Low Power Electron. Design (ISLPED)*, Aug. 2001, pp. 129–134.
[35] *Calculating Memory System Power for DDR3*, Micron Technol., Boise, ID, USA, 2007.
[36] *Monsoon Solutions Power Measurement Platform*. Accessed: Oct. 2018. [Online].Available: http://www.msoon.com/LabEquipment/_PowerMonitor/



ZONGWEI ZHU received the M.S. and Ph.D. degrees in computer science from the University of Science and Technology of China (USTC), in 2011 and 2014, respectively. From 2014 to 2016, he was a Research Assistant at the IOT Perception Mine Research Center, China University of Mining and Technology. From 2016 to 2018, he was a Senior Engineer with Huawei Company. He is currently a Research Assistant with the Suzhou Institute, USTC. His research focuses on resource scheduling, memory, power, and operating systems.



JING CAO received the B.S. degree from the College of Biomedical Engineering and Instrument Science, Zhejiang University, in 2018. She is currently pursuing the M.S. degree with the Department of Software Engineering, University of Science and Technology of China. Her current research interests include power, operating systems, artificial intelligence, and resource scheduling.

XI LI is currently a Professor of computer science and an Executive Dean of the School of Software Engineering, University of Science and Technology of China, where he directs research programes in the Embedded System Lab, examining various aspects of embedded system with focus on reliability, performance, availability, flexibility, and energy efficiency. He has led several national key projects in China and several national 863 projects and NSFC projects. He is a member of the ACM and a Senior Member of CCF.

JUNNENG ZHANG received the B.S. and Ph.D. degrees from the School of Computer Science, University of Science and Technology of China, in 2009 and 2014, respectively. He is currently a Lecturer with the School of Computer Engineering, Qingdao University of Technology. His research interests include system architecture and reconfigurable computing.

YOUQING XU received the M.S. degree from the Department of Software Engineering, University of Science and Technology of China, in 2014. From 2014 to 2015, he participated in the development of a software-defined network switch at Meshsr. From 2016 to 2018, he worked in the development of voice recognition at Q Dreamer. He is currently a Researcher and a Developer in artificial intelligence. His research interests include embedded systems, operating systems, device drivers, artificial intelligence, and deep learning.

GANGYONG JIA received the Ph.D. degree from the Department of Computer Science, University of Science and Technology of China, Hefei, China, in 2013. He has served as a reviewer of microprocessors and microsystems. He is currently an Assistant Professor with the Department of Computer Science, Hangzhou Dianzi University, China. His current research interests are IoT, cloud computing, edge computing, power management, and operating systems.