Reducing Energy of DRAM/Flash Memory System by OS-controlled Data Refresh

Vasily G. Moshnyaga
Dept. Electronics Engineering and Computer Science
Fukuoka University,
Fukuoka, Japan

Hua Vo, Glenn Reinman, Miodrag Potkonjak
Computer Science Department
University of California, Los Angeles
Los Angeles, USA

Abstract—**This paper presents a new approach to reduce energy consumption of DRAM/flash memory system by lowering the frequency of DRAM refreshes. The approach is based on two ideas: (1) a DRAM based swap-cache that reduces the number of writes to the flash memory by keeping dirty pages as long as possible; and (2) OS-controlled page allocation/aging policy that stops refreshing for banks, whose pages are clean and not accessed for a long time. Simulations show that the approach can reduce the DRAM refresh energy by 59-74% and the overall energy of DRAM/flash memory system by 8-24% without increase in the execution**

I. INTRODUCTION

Modern cell phones relay on DRAM/flash memory system to satisfy demands on fast and large data storage**.** In 2.5-3G phones, flash (mainly NAND) memory stores OS, application programs and data [1]. During boot time, the OS and program codes are copied from the flash to the DRAM in a process referred to as "memory shadowing". To reduce both the program download delay and the DRAM size, recent systems employ "demand paging" that swaps pages of code/ data between the memories on processor's requests. Such memory organization leads to a smaller DRAM, less loading time, but requires specific memory management to ensure both high speed and low energy page swapping.

II. BACKGROUND AND RELATED RESEARCH

Several issues affect page swapping in DRAM/flash memory system. The first one is *flash cleaning*, i.e. necessity to erase data before it can be overwritten. Cleaning is usually done on a *block* basis, while writing on a *page* basis. If the block size is larger than the size of transfer unit, any data that is still needed must be moved elsewhere. The next issue is *performanc*e. The time to erase and write a page in flash is 8 times longer than the time of read [2]. To avoid delaying writes for erasure it is important to keep a pool of erased memory available. The third issue is *power*. Erasing a page in NAND memory consumes twice as more power as reading the page. Since a write with erase is almost 10 times dissipates more power than a read [2], page swapping policy must minimize the number of flash memory writes, even though it might incur additional read operations. The forth issue of DRAM/flash memory system is *inactivity*. In cell-phones, a relatively short burst of operations is usually followed by long periods of inactivity during which the device is kept on standby. While all other components can be powered off on idling, the data stored in DRAM must be always refreshed or re-written to compensate the charge leakage in memory cells. Although today's DRAM chips employ a number of power down modes (e.g. active, idle (or standby) and self-refresh [3]), none of these modes appear to disable a chip entirely; refresh is always present. When device idles for a long time, the DRAM refresh may take almost 20% of the energy consumed by phone over a day [4, 5]. Clearly, reducing energy consumption of DRAM refresh can significantly extend battery life.

Research on low power DRAM refresh has been extensively performed in the past with most efforts put at circuit optimization. At the architecture level, a common approach is to exploit different retention-time among cells in DRAM imposed by temperature and process variations [6-8]. These differences provide an opportunity to save power by refreshing different cells at different rate. The selective refresh scheme [8] adds a valid bit to each memory row and refreshes the rows with valid bits set [8]. The variable refresh scheme allocates a refresh counter to each row. When the number of cycles between the previous refresh exceeds the pre-defined threshold, the line is refreshed. The retention-aware page allocation [9] favors longer-retention pages over shorter retention pages when allocating DRAM pages. This allows selecting a single refresh period that depends on the shortest-retention page among populated pages, instead of the shortest-retention page overall.

In this paper, we propose an OS-based approach to reduce energy consumption of DRAM/flash memory system. Several researches have already exploited OS support to switch DRAM banks to low power modes. Lebeck, et al [10] proposed power-aware page allocation polices that maximize use of lower power states while minimizing performance overhead of transitioning back to active mode. Fan, et al [11] studied policies for manipulating DRAM power states in cache-based systems. They showed that an immediate transition of idling DRAM chip to a lower power state might work better than a more sophisticated policy that tries to predict idling time. Lee and Chang [5] introduced an energy-aware memory allocation in heterogeneous memory systems to maximize the battery life. Delauz, et al [12] suggested various threshold predictors to determine idling time after which the DRAM should transition to a low-power state. Park, et al [13], proposed a Clean-First-LRU page allocation policy, that swaps *clean* (i.e. unmodified) pages first, while keeping dirty pages in the DRAM as long as possible. If there are no clean pages in a predefined time window, a standard LRU is used. To further reduce the number of flush memory write, Park, et al [14] proposed to combine the Clean-first-LRU with selective compression.

Despite differences the above OS-based methods have one feature in common. At the best case, they consider data retention energy of idling but do not provide means to reduce it. Our work is unique in that it focuses on OS-based policy to reduce the energy of

The work was sponsored by the Ministry of Education, Technology Science, Sports and Culture of Japan

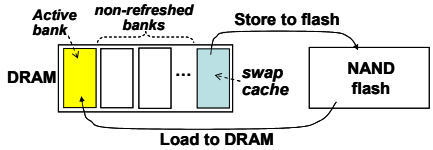


Figure 1. An illustration of proposed approach

DRAM refresh in application specific DRAM/flash memory system. As in [10, 12, 13, 5], we use history of memory accesses for energy-aware page allocation. However, unlike the prior work, we optimize allocation policy to reduce not only active energy but also the DRAM refresh energy in the flash-DRAM based memory system.

III. THE PROPOSED APPROACH

A. Main idea

The approach we propose is based on an observation that as DRAM size increases, more and more memory becomes unused at any given time. Because unused memory does not need to be refreshed, we can save energy by intelligently controlling which pages get refreshed. The system OS knows which pages are used and unused, so given the opportunity it could disable refresh on selected pages.

The main idea of our approach is simple and consists in disabling from refresh operations all individual banks which have not been referenced in given time-window *and* have no dirty (or modified) pages. If a non-referenced bank has dirty pages, we move the pages to the swap cache (see Fig.1) to keep them in DRAM as long as possible and thus minimize the number of writes to the flash storage. The swapping takes place either when the cache becomes full (in this case the LRU dirty page is moved from the cache to flash), or when the requested page is not in DRAM (in this case, the page is loaded from the flash bank to active DRAM bank).

In our approach, we exploit the fact that the OS not only has physical page allocation information for each executing process but also has information of pages that are actually being referenced (by sampling the reference bits in the page table and TLB). By compacting physical pages into minimum number of memory banks (using page coloring algorithm), we potentially eliminate refresh for entire DRAM banks in which there are no dirty pages. Modern memory systems swap out pages when the memory space is full. In our refresh-oriented page allocation, the OS starts swapping out pages when writing a page to the flash memory becomes less energy consuming than keeping the page refreshed in DRAM.

B. Assumptions

We take the following assumptions:

1. Each DRAM bank can be in two modes: *refresh* and *non-refresh* (i.e. power down). The refresh mode can be further divided into several modes, e.g. active, standby, nap and power-down as in conventional DRAM, however, we do not address this issue for the simplicity of explanation.

2. The banks are controlled separately, so each bank can be refreshed (if it in the refresh mode) or shut down (i.e. non-refreshed) independently of the others.

3. Each page can be either active (i.e. open) or closed. Any access to a close page makes it active. Refresh banks may have as open as close pages, but non-refresh banks have no active pages (all their pages are closed).

4. The higher order banks in DRAM are allocated for flash cache to minimize the number of writes to the flash memory. Dirty pages are kept in the cache until it becomes full.

C. Algorithm

The proposed refresh-oriented page allocation scheme implements the following algorithm. After a given period of time, *t1*, it detects pages, which have not been accessed and closes them. Next, after a time, *t2*, the algorithm checks status of refreshed banks. If all pages in a bank are closed, the bank is put into a non-refresh mode, while dirty pages of this bank are moved to the flash cache. Finally, after a time, *t3*, the algorithm determines the least-recently used page in the swap-cache and moves it out onto the flash-memory. After dropping the content to flash memory, the cache page is considered empty, and hence can be used to store other dirty pages. If a requested page resides in the flash and DRAM is full, the algorithm applies "clean-page-first" [13] policy to allocate the DRAM page to be swapped with the requested page. If there are no clean pages in DRAM, the algorithm moves the LRU dirty page from DRAM to the swap-cache. The code below shows the algorithm in details.

```
Algorithm:
Initialization: all banks are in refresh mode, the last bank is the
flash cache; all pages in DRAM are inactive;
At t = t2,
 find minimum number of memory banks that allocate all active
pages; move other banks to non-refresh mode;
for each access to a page AP
if page is in DRAM, access AP directly;
    else
   {find an inactive page (RP) in (non-cache) DRAM refresh-
banks;
   if no inactive pages then
      if there is a non-refresh bank, then move it to refresh
        mode;(RP = the first page in this bank);
     else
     { find CFLRU page (CFP),
       if CFP is dirty, then move CFP to cache,
         load demand page to RP (or CFP) page   }
  }
do page & bank aging
 end
move page to cache:
       find an empty page (EP) in cache;
       if EP is not found
            find a non-refresh bank (NRB) in DRAM;
            if NRB exists, then move NRB to cache;
                    let EP be first page in this bank;
            Else
               {find LRU cache page (LP) ;
                 store content of LP to the flash memory;
                   empty page LP;     }
     drop page to empty page EP (or LP);
Page & bank aging:
for each cache bank do
  {for each cache page do
    {if page is not accessed in t3, drop page to flash, empty page;
     else if a cache bank has only empty pages
     then move the bank to the non-refresh mode}}
for each refresh non-cache bank do
   {for each refresh page (RP) do
      {if RP is not accessed in t1
      then if RP is dirty, then move RP to cache;
                    move RP to the inactive mode;
      if bank has no active pages, move it to the non-refresh mode }
         }
```

IV. EXPERIMENTAL EVALUATION

A. Energy Modeling

The energy consumed by memory system is modeled by the sum of energies consumed by DRAM and flash memory: $E_{total}= E_{DRAM}+E_{flash.}$ The energy of each DRAM bank is directly proportional to the number of reads (N_{read}) and writes (N_{write}) and the unit access energy per read (E_{read}) and write (E_{write}), respectively. Furthermore, DRAM consumes idle power (P_{idle}) and refresh power ($P_{refresh}$) power during program execution. When DRAM is inactive, it stays in the power down state consuming only retention power ($P_{retention}$). Thus, assuming that DRAM consists of N banks, the energy consumed by DRAM can be calculated by,

$$E_{DRAM}=\Sigma_{i=1}^{i=N}\{E_{read}*N_{read}+E_{write}*N_{write}+t_{active}*(P_{idle}+P_{refresh}) +t_{inactive}*P_{retention}\}_i \quad (1)$$

Similarly, the energy consumption of flash memory is modeled as,

$$E_{flash} = E_{fread}*N_{fread}+(E_{fwrite}+ E_{erase})*N_{fwrite}, \quad (2)$$

where, E_{fread} and E_{fwrite} are values of energy consumed by flash per read, write and erase operation, respectively, $N_{fread,}$ N_{fwrite} are the number of flash reads and writes, respectively.

B. Experimental setup

To collect data we augmented the SimpleScalar simulator [16] with our DRAM simulation program. The SimpleScalar simulated a 400MHz 32-bit RISC processor (similar to StrongARM-110 [17]), with 32 set-associative caches (16KB inst. and 16KB data), 32B cache block size, 1 clock cycle cache hit and 3 clock-cycle cache miss, 5 clock-cycle instruction miss-prediction penalty.

Four DRAM sizes (4, 8, 16, 32, 64, and 128) MB, respectively, have been tested. The flash memory was 32MB. The energy parameters of DRAM and flash memory are taken from [2] and [15], respectively. We assumed that DRAM has 8 banks, page has 4KB, and refresh is performed every 15.6μsec per row.

Five benchmark programs (Table I) have been used in the experiment: *gcc* from the SPEC2000 suite and the others from MediaBench [18]. To model user interactions, we ran each video program 10 times with a 30 second-gap between the runs. The input video contained 100 frames and no delay between the frames. Each program was run to completion. The results have been measured in terms of the total energy consumed by the memory system, the DRAM refresh energy and the total execution time. The energy consumption of L1 (D- and I-) caches and the energy of MMU have not been considered. Also, it was assumed that OS consumes 16MB and this amount of memory was not available to the application programs. Therefore, the memory size, which the applications could freely use, was limited to 16MB unless otherwise explicitly stated.

C. Results

Figure 2 shows the breakdown in energy consumption and the execution time achieved by the proposed approach on *gcc* benchmark and normalized to conventional method [13]. Based on the results obtained at fixed *t2*, *t3* and variable *t1* (see Fig.2,a), we conclude that a small *t1* increases both the energy and the delay due to frequent page closing/opening and mode changing. Also, due to small size of DRAM, the amount of page swapping between DRAM and flash memory is large, so the flash access energy is high. As *t1* increases, both the number of page mode changes and page swapping decreasing; so the total energy also goes down. According to the results, the best value of t1 ranges between 1625ms and 3200ms. The rightmost point represents the case when *t1=t3*. As we fix *t1* at 3250ms, and vary *t2*, we see that the smaller *t2*, the better (see Fig.2, b). At 0.0125ms, for example, the refresh energy can be as much as 10%. Finally, as we expected, small t3 leads to fast page aging which extra page swapping and so increase of both DRAM access energy and flash energy. For t3 larger than 375ms, the figures do not change (see Fig.2,c).

TABLE I. BENCHMARKS AND DESCRIPTIONS

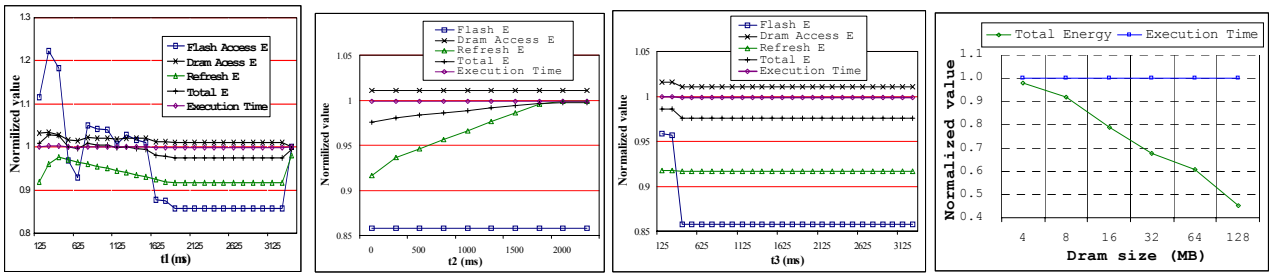
Program	Description	Symbol	Instr.x10^6
Mpeg_dec	A Mpeg2 video decoding	Mc	62
Mpeg_enc	A Mpeg2 video encoding	Md	667
jpeg_com	JPEG image compression	Jc	577
jpeg_dec	JPEGimage decompression	Jd	48
gcc	C code compiler	gcc	1,497

Figure 2(d) shows the impact of DRAM size on energy and execution time. During execution, the *gcc* program accesses 2196 different pages, which require little more than 8MB of DRAM. When the DRAM size is small, the refresh energy reduction achieved by our approach is diminished by energy consumed on page swapping. Therefore, the energy savings are small when memory is 4MB and 8MB. As memory grows, more energy can be saved. At 128MB DRAM, for example, the proposed technique can save up to 55% of the total energy without affecting the execution time.

Figure 3 shows the performance of proposed technique in perspective to the related (conventional) technique [13] for 8- and 16-MB DRAM. In this figure, bars marked by *C* show results of [13]; bars marked by *P* depict results of the proposed technique. Letters in parentheses denote the tested programs. We observe that the proposed technique lowers the refresh energy, while leaving the other energy components almost unchanged. Due to large variation in the number of pages accessed in DRAM, the results along the benchmarks as well memory size. We see that the proposed technique over performs the conventional method on all the benchmarks. Table 2 summarizes the results (Fig.4) in terms of energy reduction achieved by the proposed approach. The larger the memory size, the larger reduction ratio. For 16MB DRAM, for example, our approach reduces the DRAM refresh energy by 59-74% while lowering the total energy consumed by the tested applications by 8-26%. The maximum delay overhead observed for the benchmark was very small (less than 1%).

V. CONCLUSIONS

In this paper we proposed a refresh-driven page allocation approach to lower energy consumption of DRAM/flush memory system in portable electronic devices. According to experiments, the proposed approach can reduce the DRAM refresh energy by 74% and total energy of DRAM/flash memory by 8-26% on standard image and video applications without affecting program execution time. In this preliminary work, we have not considered the energy overhead of busses as well as the energy consumption of OS and MMU. We also acknowledge that the investigation has been restricted to benchmarks which lightly represent real handheld applications. To evaluate the approach on tasks such as internet browsing, MS-word, MS PowerPoint, Adobe Acrobat Reader, etc. an extensive profiling of the applications is needed. This work will be conducted in the near future.



(a) (b) (c) (d)

Figure 2 (a-d): Dependence of the results on t1, t2, t3, and the DRAM size, respectively

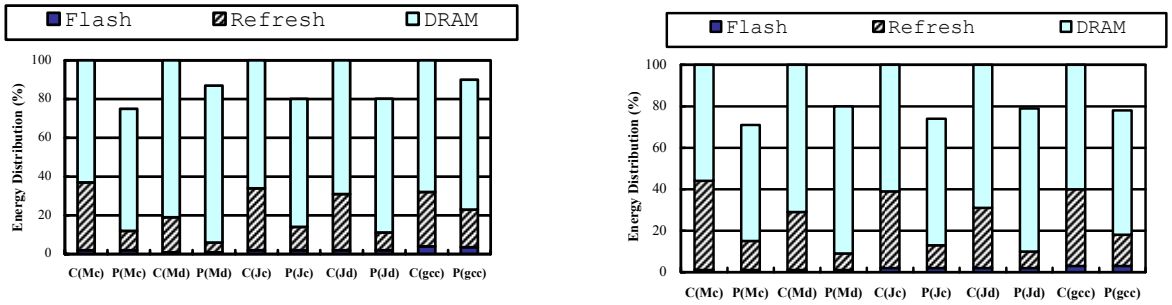


Figure 3: Results obtained for 8MB DRAM (left) and 16MB DRAM (right)

TABLE II. ENERGY REDUCTION RATIO OBSERVED FOR BENCHMARKS

Benchmarks	**Jc**		**Jd**		**Mc**		**Md**		**gcc**	
DRAM size (MB)	8	16	8	16	8	16	8	16	8	16
Refresh energy (%)	62	71	64	72	68	70	72	74	32	59
Total energy (%)	20	26	21	23	24	28	14	20	10	21

REFERENCES

[1] Weldon, T., Memory subsystems for 2.5G cellular handsets, Micron Techn. Inc., Jedex, San Jose, 2004. http://www.micron.com/products/dram/ddr2sdram/presentation.html

[2] Samsung Electronics. NAND flash memory & SmartMedia data book, 2002.

[3] Various Methods of DRAM refresh, Technical Note TN-04-30, Micron Technology Inc., 1999.

[4] Vargas, Minimum power consumption in mobile phone memory systems, Portable Design, 2006.

[5] Lee, H.G, and Chang, N., Low-energy heterogeneous non-volatile memory systems for mobile systems, J. of Low-Power Electronics, Vol.1, no.1, pp.52-62, 2005.

[6] K. Yanagisawa. Semiconductor Memory. US Patent 4,736,344, April 1988

[7] Kim,J., Papaefthymiou,M., Block-based multiperiod dynamic memory Design for Low Data-Retention Power, IEEE Trans. VLSI Systems, Vol. 11, No. 6, Dec. 2003, pp.1006-1018.

[8] Ohsawa, T., Kai, K., Murakami, K., Optimizing the DRAM refresh count for merged DRAM/logic LSIs, ACM/IEEE ISLPED 1998, 82-87.

[9] Venkatesan, R., K., Herr, S., Rotenberg, E. Retention-Aware Placement in DRAM (RAPID): Software Methods for Quasi-Non-Volatile DRAM, Proc. IEEE HPCA-12, 2006

[10] Lebeck, A.R., Fan, X., Zeng, H., Ellis, C., Power-aware page allocation, Proc. 9th Int. Conf. on Architectural Support for Programing Languages and OS (ASPLOS IX), Nov. 2000

[11] Fan, X., Zeng, H., Ellis, C., Lebeck, A.R., Memory controller policies for DRAM power management, Proc.ACM/IEEE ISLPED, 2001.

[12] Delauz, V., et al., Scheduler-based DRAM energy power management, 39th ACM/IEEE DAC, pp.697-702, 2002.

[13] Park, C., Kang, J.U., Park, S.,Y., Kim, J.S., Energy aware demand paging on NAND flash-based embedded systems, Proc. ACM/IEEE ISLPED-2004, pp.338-343.

[14] Park, S., Lim, H., Chang, H., Sung, W., Compressed swapping or NAND flash memory based embedded systems, Proc. IEEE Workshop on Signal Processing Systems (SiPS), 2003.

[15] Samsung Electronics, 128Mb DDR SDRAM Specification, Version 1.0, Rev.1.0, Nov.2, 2000.

[16] Burger D., Austin, T., Bennet, S., Evaluating future microprocessors-the superscalar tool set. Technical Report 1306, Univ. of Wisconsin-Madison, CSD, July 1996.

[17] SA-110 Microprocessor, Technical Reference Manual, Intel Corporation, Dec.2000.

[18] C. Lee, M. Potkonjak, and W.H. Mangione-Smith, MediaBench: a tool for evaluating and synthesizing multimedia and communication systems, Proc. the IEEE Int. Symp. on Microarchitecture, 1997.