# EXTREME: Exploiting Page Table for Reducing Refresh Power of 3D-Stacked DRAM Memory

Ho Hyun Shin, *Member, IEEE*, Young Min Park, Duheon Choi, Byoung Jin Kim, Dae-Hyung Cho, and Eui-Young Chung, *Member, IEEE*

**Abstract**—For future exascale computing systems, ultra-high-density memories would be required that consume low power to process massive data. Of the various memory devices, 3D-stacked DRAMs using TSVs are a perfect solution for this purposes. In addition to providing high capacity, these provide functional flexibility to the computing system by attaching a logic die in each 3D-stacked DRAM chip. However, the high capacity 3D-stacked DRAMs suffer from a significant loss of refresh power, which is solely required to maintain data. Although various schemes have been proposed to mitigate this issue, they cannot be adopted by commercial products due to compatibility issues. To tackle this issue, we propose EXTREME, which effectively reduces the refresh power of 3D-Stacked DRAMs. In order to retain the compatibility with OS, a simple page table manager is implemented at the logic die of 3D-stacked DRAM devices, which pins the page table to a specific memory space. The experiment results demonstrate that this reduces the refresh power at idle time by up to 98 percent with 16 KB of SRAM (static RAM) and 64 KB of DRAM register overhead for a 2 GB 3D-stacked DRAM device.

**Index Terms**—DRAM, low-power design, 3D-stack, refresh, page table

## 1 Introduction

Modern computing systems have evolved to contain multi-core processors and high density memories for the big data era. In particular, server systems require considerably parallelized processors and ultra-high-density memories to support big data processing. To this end, CPU manufacturers focus on improving clock frequency and maximizing the number of cores. On the other hand, DRAM vendors put in a great deal of effort for increasing DRAM capacity at low cost. While the technological enhancements considerably improved the data processing capability, they also rapidly increased the power consumed by the entire system. The power of the DRAM device in particular is rapidly growing due to capacity increase, and a recent study have shown that it accounts for up to 26 percent of the overall system power [1].

The operating power of the DRAM chips is mostly affected by the operating bandwidth and voltage. However, the refresh power, which is only needed to retain data, is directly related to the capacity. This is because all DRAM cells must be refreshed at least once in a refresh period(typically 64 ms). Fig. 1 shows the refresh and non-refresh power consumptions of the various DRAM densities. It depicts that the refresh power is increased proportional to the density and is expected to account for 47 percent of total device power in a future 64 Gb(gigabit) DRAM device. This graph indicates that the refresh power of DRAM can no longer be ignored and it will become one of the critical design parameters in the future.

To support high capacity memory, 3D die stacking technology using through silicon vias(TSVs) has emerged. It stacks several DRAM dies in a chip by interconnecting them with hundreds of TSVs for boosting capacity. Not only can the 3D-stacked DRAM device maximize memory capacity, it can also provide a wide range of design flexibility to the computing system. For example, a 3D-stacked DRAM device, which incorporates a logic die, can achieve maximized access bandwidth and low power usage at the same time. High bandwidth memory(HBM) and hybrid memory cube(HMC) are examples of the adoption of these advantages [3], [4]. In addition, these examples can implement various functional circuits in the logic die such as built-in self-test(BIST) and processing in memory(PIM) [5]. Although the 3D-stacked DRAMs have these various advantages, they are suffering from heat dissipation problems occurring in the chip itself. It is the most important factor that reduces the refresh cycle of DRAM and increases its power.

Many previous studies have proposed various refresh power reduction schemes at the OS level. Some of them have optimized DRAM sub-systems for eliminating unnecessary refresh operations, while others have implemented selective refresh mechanisms by modifying overall system architecture. However, due to the lack of compatibility with modern computing systems, they could not be adopted as commercial products. We discuss these issues in detail in Section 3. In this paper, we propose a new refresh power

- *H.H. Shin is with Samsung Electronics Company, Ltd., Hwasung 18448, Korea, and with Yonsei University, Seoul 03722, Korea. E-mail: hhshin@dtl.yonsei.ac.kr.*
- *Y.M. Park, D. Choi, B.J. Kim, D.-H. Cho, and E.-Y. Chung is with the School of Electrical and Electronic Engineering, Yonsei University, Seoul 03722, Korea. E-mail: {ympark0225, cdh0527, bryankim, dhcho, eychung}@yonsei.ac.kr.*

*Manuscript received 29 Dec. 2016; revised 25 June 2017; accepted 28 June 2017. Date of publication 3 July 2017; date of current version 19 Dec. 2017. (Corresponding author: Eui-Young Chung.) Recommended for acceptance by S.-Y. Huang. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TC.2017.2723392*

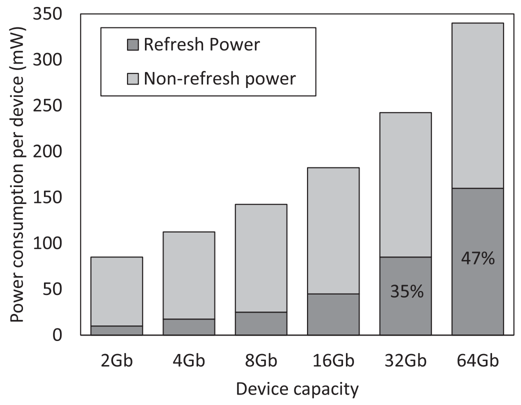


Fig. 1. As the DRAM memory capacity increases, the refresh power increases proportionally [2]. This refresh power occupies up to 35 and 47 percent of the total DRAM power of 32 and 64 Gb devices, respectively.

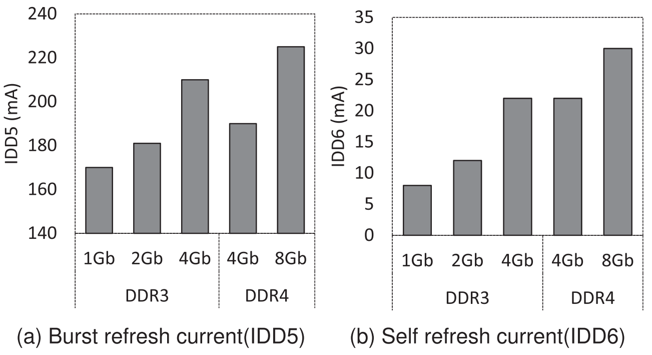


Fig. 2. Comparison of refresh current for DRAM devices of various capacities. [6], [7]. The refresh current is linearly related to the DRAM cell capacity because as the DRAM capacity increases the number of cells to be refreshed increases in a fixed refresh period. The normal operating for of DDR3 and DDR4 DRAMs are 1.5 and 1.2 V, respectively, which creates a difference in the two refresh current values.

reduction scheme called EXTREME(exploiting page table for reducing refresh power of 3D-stacked DRAM memory) to resolve the compatibility issues. It pins a specific physical memory space for the page table of the OS. In addition, this approach only requires a minor change to the OS and 3D-stacked DRAM without any modification in overall system architecture.

In this work, we mainly concentrate on how to handle the page table to pin it into specific DRAM space and how to manage the information to control refresh operations with minimum cost overhead. The main contributions of our work are described as follows:

(1) *Pinning the page table in a specific physical address space.* We study the page table management of the OS and introduce a simple way to pin the page table into a specific physical address. It causes the page table to be located in the specific area defined by the user without modifications of the processor and DRAM controller.
(2) *Hardware design for the refresh management of the 3D-stacked DRAM device.* We implement the hardware resources for the refresh management in a logic die of the 3D-stacked DRAM device. This refresh manager is in charge of ensuring that the page table is compatible with the OS. The proposed scheme requires marginal overhead, which is 16 KB of SRAM and 64 KB of DRAM registers for a 2 GB(gigabyte) 3D-stacked DRAM device.

In Section 2, we summarize the basic refresh operations of DRAM and the concept of a page table. In addition, we address the various refresh power reduction schemes that have been introduced and explore their advantages and disadvantages in Section 3. In Section 4, we suggest how to implement EXTREME in terms of software and hardware. Section 5 shows the simulation results of EXTREME. Finally, we conclude the proposed scheme and discuss future work in Section 6.

## 2 Background

In this section, we skip the basic concept of DRAM refresh. Instead, we explain why the refresh power is critical as its capacity grows, and we compare two refresh interface policies. In addition, details of the page table walk of the Linux OS are explained in order to provide fundamental knowledge related to EXTREME mechanism.

### 2.1 DRAM Refresh

All rows in a DRAM device should be refreshed at least once in a refresh period($tREF$ = 64 ms). In order to minimize the side effects of the refresh operation on normal operation, the number of refresh commands in a $tREF$ is fixed to a specific value by a DRAM controller. Typically, it is set to 8,192(8K) at room temperature. However, the number of rows to be refreshed for a single refresh command is increased as the capacity of a DRAM device increases. This means that, in order to refresh all cells in a refresh period, multiple word lines(WLs) should be refreshed simultaneously. For example, an 1 GB DRAM device which has a total of 128K WLs should refresh 16 WLs in a refresh command, while a 4 GB DRAM device should refresh 64 WLs at the same time. This example explains why high capacity DRAM devices consume much more power than the low density devices. Fig. 2 shows refresh current per DRAM chip with respect to various cell densities. Fig. 2a shows the burst refresh current (IDD5), which is consumed in burst auto refresh operation mode. In contrast, Fig. 2b shows the self-refresh current (IDD6) measured at power down mode. In the self-refresh mode, the DRAM controller keeps all normal operations from stopping and the DRAM device operates the refresh by itself. As can be seen from Fig. 2, the refresh power is linearly increased according to capacity even in power down mode.

By default, a DRAM controller is responsible for the refresh operation, and thus it periodically generates refresh commands according to the retention time of the DRAM device. Along with the command, the row addresses to be refreshed should be asserted by the DRAM controller or generated by the DRAM device itself. The former and the latter policies are referred to as ras-only-refresh(ROR) and cas-before-ras(CBR) respectively. While classical asynchronous DRAMs adopted ROR refresh policy, modern synchronous DRAMs adopt CBR as a standard policy for its convenience and power efficiency. Fig. 3 compares the differences between the two policies. Since the DRAM controller adopting CBR policy does not need to equip a refresh counter, it simplifies the controller. In addition, as the refresh counter is incorporated in the DRAM device instead

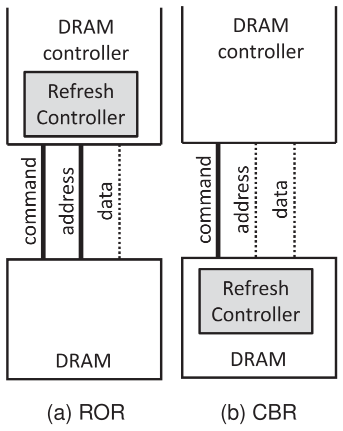


Fig. 3. Conceptual comparison of ROR and CBR refresh policies. (a) The ROR policy has a refresh counter on the DRAM controller and uses the command and address interfaces. (b) By contrast, CBR has a refresh counter on the DRAM device and uses only the command interface.

of the controller, the power dissipated by address signal transmissions can also be saved.

## 2.2 Page Table Management of OS

To enhance memory space utilization, the processor converts the virtual addresses used in the software into the physical addresses. Typically, the OS and memory management unit (MMU) in a CPU are responsible for the translation by referencing the page table. The page table resides in the main memory or the cache when a process is invoked, and is removed when it is terminated. Once the page table is created by the OS, MMU takes the role of the translation for the invoked process. By default, the page table resides in the memory sub-system. Therefore, whenever the CPU accesses the memory with a virtual address, it needs to find a physical frame number(PFN) corresponding to the address through the page table walk. However, the translation process can be burdensome because of the long memory latency. In order to resolve this problem, most modern computers are equipped with a translation lookaside buffer(TLB) in the MMU [8]. A TLB is similar to a cache and temporarily stores the page table entries. Fig. 4 shows the translation procedure from a virtual address to a physical one with a TLB. When the CPU accesses the memory with a virtual address, the MMU finds the corresponding physical address in the TLB. If it finds the address, it carries out the translation immediately. In contrast, when the TLB does not include the address, the MMU searches for it in the cache or in the main memory.

Modern computing systems have a multi-level page table translation hierarchy. For example, an ARM system has a two-level hierarchy for maximizing memory utilization[1] [8]. Fig. 5 shows a two-level page table walk in an ARM system. Each process has a translation table base address(TTB) which is stored in a register of the MMU. When a process tries to access memory with a virtual address, the MMU first refers to the TTB and translates the virtual address into the first level page table address. Here, the first level page table is called by the page directory in the Linux kernel. The TTB

1. In this paper we explain the scheme with ARM system reference. However, because the virtual to physical translation process is almost similar to other systems it can be adopted to others flexibly.

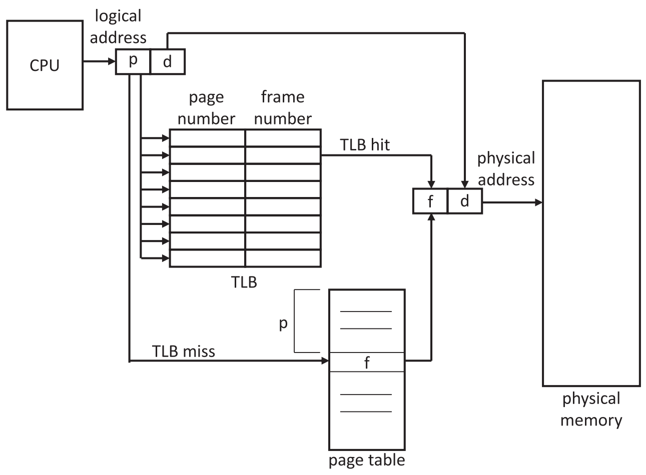


Fig. 4. Translation process from virtual address to physical address. [9]. The TLB acts as a cache for virtual-to-physical translations. However, the page tables are primarily present in the main memory, such as caches or DRAMs.

is the base address of the page directory and the twelve most significant bits(MSBs) of the virtual address are indexed. The entry corresponding to the first level address has a descriptor, which defines the configuration of the page. The second page walk is similar to the first one. Finally, the physical address is generated by the page base address stored in second level page table and page index of virtual address. Because the descriptor of an entry in the second page table has twenty bits of base address and the remaining twelve bits are fulfilled with the page index, an entry covers 4 KB of the physical address. It means that if the page is freed by the kernel, the 4 KB of physical space is not necessary anymore until it is allocated again.

The point we focus on in this work is that the page table has information about the physical address location, which would be accessed by the processor. This means that if the information can be transferred to the DRAM side, the DRAM device can reduce refresh power by refreshing only valid rows that include valid data. However, the most difficult problem is how to transfer the page table into the DRAM device while maintaining compatibility. We describe the solutions in Section 4.2 and show how the mentioned structure of the page table is modified.

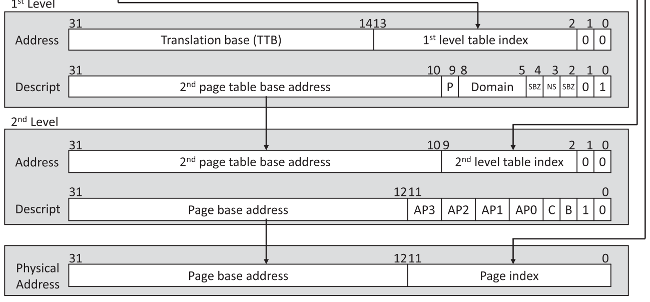


Fig. 5. Two level page table walk of ARM systems [8]. Each process invoked from a processor has a TTB for the virtual to physical address translation, and the virtual address issued by the processor are translated into physical memory address through the level one and two page table walk process.

TABLE 1
Comparison of Various Related Works to Reduce Refresh Power

| Cat. | Scheme | Modifications | Refresh Policy | Storage Overhead (for 4 GB) |
|---|---|---|---|---|
| (1) | SMART REFRESH [10] | DRAM controller | ROR | 96 KB |
| (2) | VRA [11] | CPU & OS DRAM controller DRAM device | ROR | <512 KB |
| | RAPID [12] | OS | ROR | 4 MB |
| | FLIKKER [13] | OS | CBR | - |
| | RAIDR [2] | OS DRAM controller | ROR | 1 KB |
| | RIO [14] | OS | CBR | - |
| (3) | SRA [11] | CPU & OS DRAM controller DRAM device | ROR | <512 KB |
| | ESMIKO [15] | CPU & OS DRAM controller DRAM device | ROR | 472 KB |
| | PARIS [14] | OS DRAM controller | ROR | 1 KB |
| | PASR [16] | OS DRAM device | CBR | - |
| | EXTREME | OS DRAM device | CBR | 128 KB (DRAM) 24 KB (SRAM) |

*Category (1), (2) and (3) indicate the schemes described in Sections 3.1, 3.2 and 3.3, respectively.*

## 3 Related Works

This section describes various studies on methods that reduce the refresh power of DRAM devices. They can be categorized into three types according to their implementation. The first is to modify only the DRAM controller for reducing unnecessary refresh operation duplicated by activation commands. The second is to take advantage of the DRAM cells that have longer retention time than the DRAM specification. The last is to use the memory space activated by the OS or application. Table 1 gives an overview of various related works. This section discusses the details of each category and their advantages and disadvantages.

### 3.1 DRAM Controller Directed Method

One simple way to reduce refresh power is to make the DRAM controller smart and eliminate unnecessary refresh operations [10]. Normal activation and refresh operations are the same in terms of charging DRAM cells. Therefore, if any row is activated at any time, there is no need to refresh the row within the retention time(typically 64 ms at room temperature) from the active time. This method has the great advantage in that it can be implemented simply by modifying the DRAM controller. However, many register sizes are required to store the flag bits for every row, and the refresh power can be reduced only at runtime, not during idle times. In addition, ROR refresh policy must be accommodated instead of CBR, which is the latest DRAM subsystem standard.

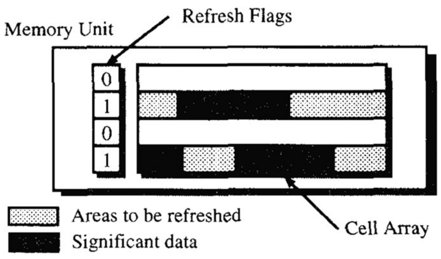


Fig. 6. An implementaion of SRA [11]. Black area contains the significant data. Therefore, the rows that include the black areas should be refreshed.

### 3.2 Exploiting DRAM Retention Time with OS Management

A DRAM device consists of huge cells with a wide retention time. However, DRAM vendors typically test and screen with a conservative single reference that can guarantee fail-free operation. This is because they try to reduce test time to achieve low cost. Due to the limited testing process, a considerable number of cells have much longer retention time than the reference point. In view of this situation, various refresh management techniques have been introduced that take advantage of retention time. The representative schemes are RAPID, FLIKKER, RAIDR and RIO [2], [12], [13], [14]. By default, these technologies require the OS to know retention time information for all DRAM rows. DRAM uses this information to selectively perform the refresh operation for each row. This methodology can effectively reduce power by increasing the average refresh time. However, due to the shrinking of DRAM process technology, cell leakage is much worse than before and the weak cell area increased sharply. This phenomenon decreases the effectiveness of these technologies. [17]. Furthermore, It is not cost effective to store the retention time information in a DRAM device, and the DRAM test of the user's system is not reliable because the test pattern cannot fully reflect the manufacturer's test procedure. As a result, although this technology can be effective at reducing refresh power, it costs a lot to apply these ideas to mass-produced products.

### 3.3 Architectural Level Changes Including OS

Since activated data is finite among all the data stored in DRAM, DRAM refresh is closely related to overall system architecture and software. Therefore, if the DRAM subsystem can know the range of activated data, the refresh operation can be selectively operated so that the refresh power can be optimized. Selective Refresh Architecture (SRA) was introduced based on this idea [11]. Fig. 6 shows the abstracted concept of this scheme, where the refresh flag indicates which rows will be refreshed or not, depending on the location of the activated data. The key to this idea is how information about the activated data is transmitted to the DRAM controller or DRAM device. However, a clear method for transmission was not proposed at that time.

Based on SRA ideas, a variety of techniques have been introduced to reduce refresh power. ESKIMO used semantic information for memory allocation and deallocation [15], and PARIS made use of the page management of the OS [14]. In order to transfer the semantic or page information to the DRAM side, however, all system architectures such as the OS, processor, and DRAM should be modified. Furthermore,

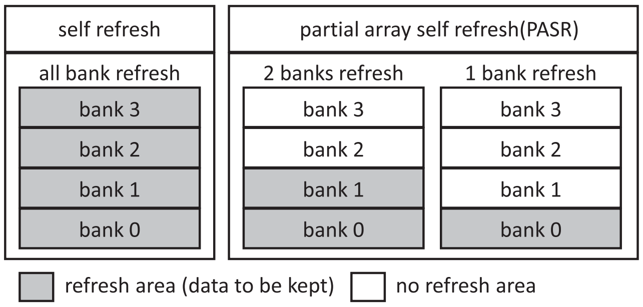


Fig. 7. Overview of PASR shceme [16]. While a typical self-refresh method refreshes all DRAM cell arrays, PASR refreshes only the partial regions defined by the OS.

these techniques adopt ROR refresh policy to transfer the information into DRAM devices. This is because the DRAM controller should own a refresh related register map and directly manage the refresh of the DRAM device. Even though the architectural level changes are the ultimate in reducing refresh power, they require a lot of modifications to software and hardware resources. Moreover, the ROR refresh policy is not suitable for a modern computing system.

The most practical low-power refresh scheme based on systematic management is partial array self-refresh(PASR), which is implemented on commercial low-power DRAM devices [16]. In the PASR scheme, a DRAM device receives the information about which banks or arrays are activated from the OS and only refreshes the activated rows when the DRAM enters the self-refresh mode. Fig. 7 shows the difference between normal self-refresh and PASR. The scheme has significant advantages in that it is simpler than others and does not demand large modifications to the processor or DRAM controller. However, it has some limitations. First, it can reduce refresh power solely at idle time (i.e., self-refresh mode) not at runtime. Second, the valid data rearrange process is very complex and needs timing overhead before entering self-refresh mode. Finally, the refresh power reduction granularity is restricted to a bank or specific array unit.

We summarize the schemes in Table 1. With consideration for the various schemes, we design a simple and effective refresh power reduction method. Unlike prior architectures, our proposed architecture can be implemented with modifications only to the OS and DRAM devices. Furthermore, it can adopt CBR refresh policy and does not need the DRAM retention information. In the following section, we describe the details by separating them into hardware and OS.

## 4 Proposed Method

### 4.1 Overview

The first objective to exploit the OS in the refresh power reduction area is to transmit the address information of activated data to the DRAM side. However, the process is not trivial and needs significant consideration. One of the proposed methods to resolve this is using the page table structure of the OS. If the page allocation or deallocation information can be properly transmitted to the DRAM side, a DRAM device can exploit it to manage refresh operations. However, it requires several modifications of the processor, DRAM controller, and DRAM device. First, a processor must create a new ISA and packet structure to transmit the

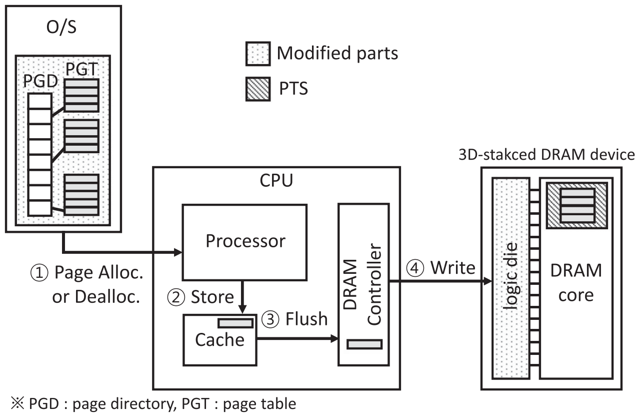


Fig. 8. Brief operation diagram of the proposed scheme. When the OS issues a write command for the page allocation, the processor sends the store packet to the memory side. When the cache receives the packet, the page must be flushed to the DRAM side to reflect page allocation immediately.

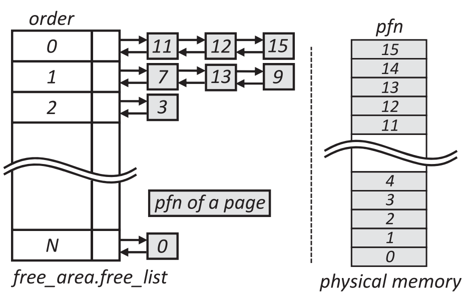
page table related information to the DRAM side. In addition, the DRAM controller should set up a new command for transmitting the information, and the DRAM device must equip a new decoder for this command. These approaches increase the design complexity and cost overhead of the overall computing system. To overcome these problems, we focused on how to deliver the pages to the DRAM side with high compatibility, and finally solved this by pinning the page table to a specific DRAM address space. This method does not require modification of the processor or DRAM controller, but only modifies the page table related OS code and the 3D-stacked DRAM device. The details of the OS implementation and hardware enhancement are described in Sections 4.2 and 4.3, respectively.

Fig. 8 depicts the overall operation flow of EXTREME. It shows the typical computing system diagram with the modifications based on the proposed scheme. First of all, the page table should be pinned to a specific physical address space in the DRAM device, which is called page table space(PTS). Here, PTS size can be calculated as shown in Eq. (1), where we assume that an entry of a page table occupies 4 bytes and that it covers 4 KB of physical address space. If the DRAM capacity is 4 GB, for example, the necessary PTS size is 4 MB and it accounts for only 0.1 percent of the total capacity

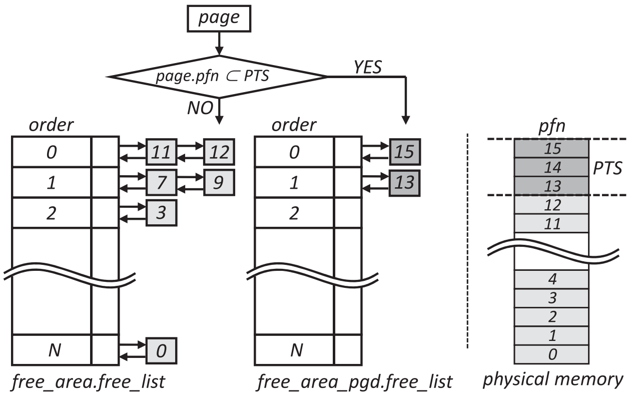
$$PTS\ size\ = \frac{DRAM\ capacity}{page\ size} \times \frac{4}{1024}(KB). \tag{1}$$

When the OS allocates or deallocates a page, it issues a store instruction with the physical address included in the PTS according to the modified OS codes(①). The processor that receives the instruction asserts a store instruction for the page to the memory side(②). When a cache receives the packet including the instruction, it flushes the page to the DRAM side in order to reflect all page allocation information to the the DRAM device(③). Finally, the DRAM controller sends a write command into the DRAM device with page related information(④). Because the address of the page table is pinned into a specific address range, it does not need to identify whether the write command is for page allocation or for normal operation.

Our proposed method has notable advantages over the prior schemes in terms of compatibility with existing computing



(a) Conventional *free_area* management



(b) Proposed *free_area* management

Fig. 9. Management of *free_area* [9]. A new *free_area* structure for the page directory is created to pin the page table to a specific memory space. Pages in the page directory are assigned to the new structure, and the system refers to the new directory for page table allocation.

system. First, the proposed scheme does not need to be concerned about the communication between the processor and the DRAM subsystem. Second, the logic die of 3D-stacked DRAM can perform various functions with the page table information, including refresh management or a page swap.

### 4.2 Software Enhancement on the OS

The most critical aspect of the EXTREME scheme is how to pin the page table structure into specific physical address space without the support of the processor or DRAM controller. Since the OS kernel runs with only virtual memory address, the pinning process is not trivial. Fortunately, however, this problem can be solved since the page table structure of the Linux kernel has physical frame related information (i.e., PFN). In addition, the buddy algorithm, which manages page table structure, provides the possibility of implementing the pinning.

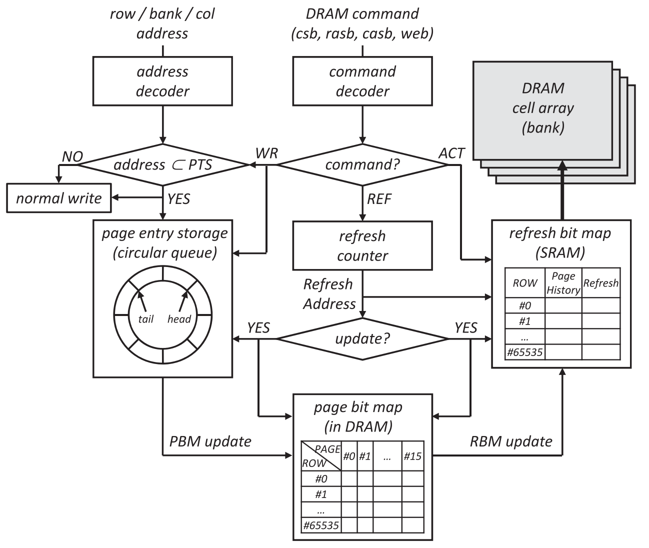
The buddy algorithm manages the page table with a *free_area* structure [9]. The structure is composed of multiple entries of order numbers. Each entry points to the memory page corresponding to the order, where the order indicates the size of the page blocks. Each entry of the *free_area* has a map of the page blocks for each order. For example, entry N of the array has a memory map that describes that are each $2^N$ pages long. Fig. 9a shows an example of the abstracted buddy system and physical memory space. When a process needs a page or several contiguous pages, it refers to the *free_area* and takes a single page or multiple pages, according to the request. In this work, we assume that the address bound of the page table space(PTS) is defined in the booting sequence and the Linux kernel can refer to it. In addition, when multiple processes are running at the same time in a system, each process manages its own page table. Even if the number of processes increases, the size of the PTS does not need to increase. This is because the range of physical memory address used by the entire processes is the same as the physical address range of the actual DRAM device, even when multiple processors are running. That is, the buddy algorithm of the OS kernel dynamically allocates pages to each process within the entire physical address range. In addition, since the PTS eventually has address information for the allocated pages, the size of the overall PTS does not need to increase.

To implement the pinning of the page table, we explored the multi-level page table walk structure of the Linux OS and found some useful characteristics. First, a page directory points to the physical base address of a page table. Second, the page directory is allocated from the *free_area* structure like the page table. These explorations give us a chance to implement the page table pinning. If we design the page directory to be allocated with pages that only point to pre-defined PTS, the page table can be within the fixed address range only. In addition, since the page directory also follows the buddy algorithm, the page table can be pinned to the PTS by handling the algorithm's code. We explain the mechanism with an example in Fig. 9b.

We add a new *free_area* structure, called *free_area_pgd*, to the buddy system for the page directory(see Fig. 9b). The new structure only links pages that have a page frame number(PFN) included in the range of the PTS. This can be achieved by comparing the PFN of the page with the PTS. Based on the separated *free_area* structure, when a page directory requests a single page or multiple pages, it refers to the new *free_area_pgd*. When a page table needs some pages, on the other hand, it fetches them from the original *free_area* structure, unlike the page directory. The new structure and its management codes make the page directory just point to the PTS and thus let the page table be pinned into a specific address range automatically.

An entry of page directory can directly allocate a large 1 or 4 MB physical page in the two-level page table walk, instead of pointing to a page table address. In this case, it is surely better to stop the refresh for the super page when the deallocation of the page is issued. However, we ignored the large size of the page deallocation by first-level page table. This is because most of the page directory entries usually point to page table addresses, and the exceptional case makes the page table related codes complex. Furthermore, even though the deallocation information of the large pages cannot be transmitted to the DRAM, it does not affect the normal operation of the proposed scheme. This is because the refresh operation is enabled by row activation and disabled by page deallocation information, which is entered into the PTS.

The implementation using the *free_area* structure has an advantage in that it is scalable to various DRAM architecture. So far, we have assumed a single chip 3D-stacked DRAM structure because the logic die in it should manage the whole physical address range. However, the OS manages the memory address with several zone spaces and each zone has a *free_area* structure. Therefore, if the OS assigns a single zone to each DRAM device that has a different channel or structure, multi-channel or multi-chip



| | | |
|---|---|---|
| Page entry storage(PES) | Register (circular queue) | Register for temporarily storing the valid memory Addresses to be refreshed. |
| Page bit map(PBM) | DRAM | DRAM storage for storing the valid memory addresses to be refreshed. |
| Refresh bit map(RBM) | Register (table) | Table that stores the bits related to the refresh progress of each row. |

Fig. 10. Operation diagram of the proposed new scheme in DRAM device. The PES and RBM are implemented to minimize the register overhead. When a page table entry is issued from the processor with a store instruction, the PES temporarily stores the data with circular queue manner and updates the PBM periodically. The bit data of the PBM also periodically reflected into the RBM and finally refresh controller of DRAM refers the RBM for smart refresh management. The table at the bottom of this figure summarizes the implementation and operation of each storage.

DRAM devices can adopt the proposed scheme without much effort.

Even though software modification can play a key role in EXTREME, the hardware implementation of the DRAM device is critical for managing refresh operations using the PTS information. Since it directly affects cost overhead of the DRAM device, in addition, simple logic circuits and small register sizes are important. In Section 4.3, we describe the hardware implementation of the 3D-stacked DRAM device in detail.

## 4.3 Hardware Enhancement on the 3D-Stacked DRAM Device

The page information transferred to the DRAM side should be used to control refresh operations. The best way to implement the refresh control logic is to create a bit map table for all page frame numbers(PFNs) with large registers or SRAM memories. Using the table, the control logic can selectively refresh the rows corresponding to the bit data stored in it. However, this method requires a significant amount of register storage. For example, 2, 4 and 8 GB DRAMs need 64, 128 and 256 KB of storage respectively. These large resistors can cause unintended leakage currents, which can eventually hinder the refresh power reduction.

In order to overcome the problems, we implement a bit map table called the page bit map(PBM) in the DRAM cells, instead of the SRAM. This is because DRAM has much lower current leakage than register or SRAM, and it is more cost effective. Furthermore, we add the two SRAM registers, which are page entry storage(PES) and refresh bit map(RBM),

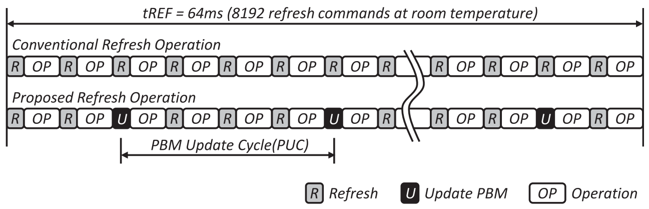

tREF = 64ms (8192 refresh commands at room temperature)
Conventional Refresh Operation
Proposed Refresh Operation
PBM Update Cycle(PUC)
R Refresh
U Update PBM
OP Operation

Fig. 11. Refresh operation according to the proposed scheme. Because the PBM is implemented in a DRAM array, some refresh cycles must be intercepted and updated periodically. Intercepted rows must be distributed over other normal refresh cycles and may cause peak power increase of the refresh operation.

to compensate the long latency of the PBM. Fig. 10 shows the operation diagram of the proposed hardware architecture.

### 4.3.1 Update the PES

When a page allocation or deallocation instruction is issued from a CPU, the DRAM controller transfers a write command to the DRAM device. The address control unit in the device checks whether the address is included in the page table space(PTS). If the address is in its address range, the address is stored in the PES and the normal DRAM cell at the same time. The PES is configured to a circular queue and its size is a critical design parameter of EXTREME. The twenty most significant bits of the stored data indicate the PFN, and lower twelve bits keep the properties of the frame. Because we design the OS to flush the cache whenever it allocates or deallocates a page entry, all page table entries are transferred to the DRAM device.

### 4.3.2 Update the PBM

The data stored in the PES should be transferred into the PBM. However, it is impossible to access the PBM at normal DRAM operation time since the PBM is implemented in the DRAM array. To resolve the problem, we intercept some refresh cycles and convert them into the time required for updating the PBM(see Fig. 11). For example, when the PBM is updated every fourth refresh command cycle, such as in Fig. 11, the rows reserved at the update cycle should be distributed among the remaining three refresh command cycles. We define the cycle for updating the PBM as the PBM update cycle(PUC) in this paper. However, it inevitably increases the peak refresh power of the remaining cycles to 25 percent. Therefore, the PUC should be maximized and the peak power should be considered in the design step. The optimal PUC will be explored in Section 5.2, which shows that only 32 PUC is enough to effectively reduce the refresh power. This means that the additional peak power can be only 3.125 percent, and it can be properly managed by various hardware design methodologies, such as strengthening the power decoupling capacitor and power-mesh layout. Apart from the peak power compensation, different techniques can be considered to manage the RBM update. For example, increasing the refresh peiord(*tREF*) can be one of these solutions. It inserts the update cycles in several refresh cycles and expands the *tREF* to the value of the inserted update cycles. However, this capability should consider the retention time increment of all DRAM cells [18]. The other technique is adding refresh counts in a *tREF* for updating the RBM. This method also has a disadvantage in that it can aggravate the bandwidth of a DRAM device.

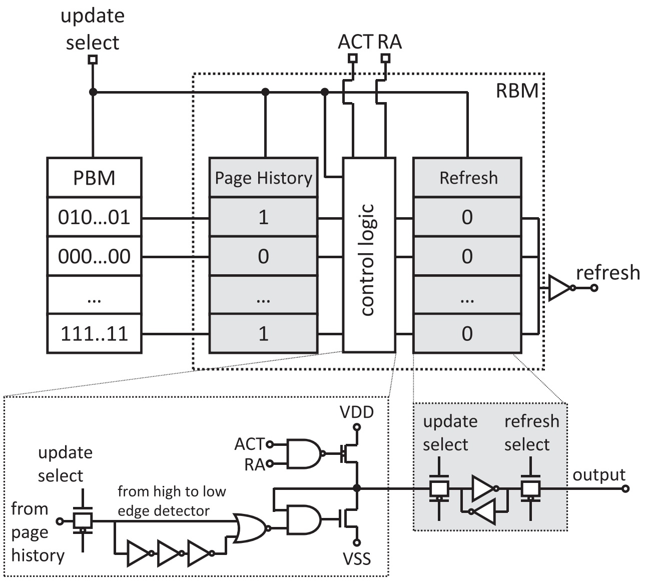


Fig. 12. Circuit diagram of RBM. For complete refresh operation, the RBM consists of two columns. The first is set by the active command and the second is controlled by the bits in the PBM.

The number of PES entries transferred into the PBM in an update cycle is also important to determine the efficiency of the proposed scheme. The minimum write or read cycle of normal DDR3 DRAM is defined as *tCCD* and is typically set to 6 ns. It means that a single column access is allowed within a *tCCD* cycle. In this case, it is needless to deeply consider row activation time because the PBM consists of only 64 KB for the 2 GB DRAM device, and it makes up only eight WLs. If we exploit bank interleaving manner, the eight WLs can be activated at once and thus only the column access needs to be considered. Because an entry of PES stores data of a page, it needs a column access to update a bit of PBM corresponding to the PES entry. Therefore, the transferred number of PES entries can be calculated as *tRFC*/*tCCD*, where *tRFC* is the refresh cycle time and defined as 160 ns for the 2 Gb DRAM device. The update time cannot be fully used for updating the PBM because some of the *tRFC* should be assigned for updating the RBM. However, the updating time of the RBM is much smaller than that of the PBM since 64 bytes(512 bits) of the RBM can be updated at the same time within a single *tCCD*.

### 4.3.3 Update the RBM

The RBM is the most critical component to the refresh operation because it stores the refresh control bit data. The map consists of several entries same to the number of rows in a bank. Typically, the OS manages the memory space with 4 KB per page unit, while a DRAM device controls the refresh operation with 8 WLs, which correspond to 64 KB due to the bank parallelism. Therefore, a row of the map indicates the status of sixteen physical pages. In addition, the RBM is composed of two columns. The first column, page history field, stores the page history, which is set by the row address of a normal active command. The second column, refresh field, stores the final information about whether a row should be refreshed or not. The reason why the first column should be implemented is to ensure the perfect refresh operation. In some cases, a page can be allocated in the page directory directly instead of in the page table, for high utilization of virtual to physical translation. When an address of a page is not included in the PTS, the physical addresses corresponding to the page should be refreshed without page allocation information. Therefore, we design the system to set the page history field with an active command, and to reset the refresh field only if the page history bit and refresh fields are both one and an inserted page history update bit is going to zero. Fig. 12 shows the circuit diagram for the set and reset procedure.

With the proposed hardware resources, EXTREME can adopt the CBR refresh policy instead of ROR. This means that it does not need the help of the DRAM controller to refresh the DRAM cells. ROR policy can more easily implement the transfer of page-related information, but is no longer used in recent DRAM subsystems. Therefore, our proposed EXTREME technology is more compatible with recent systems.

### 4.3.4 Overhead Estimation

We analized the area size of the PES and the RBM with the CACTI tool, which is an integrated cache and memory area, and a leakage model [19]. The size of the RBM is fixed with DRAM density and page size, so the RBM size can be expressed as shown in

$$RBM = \frac{Capaticy \div (Page\ Size \times 16)}{8 \times 1024} \times 2(KB). \quad (2)$$

According to this equation, the 2 GB DRAM needs an 8 KB RBM. In contrast, the PES size should be determined by the experimental results in the design space. In this paper, we selected the 8 KB PES, according to the experiment results of Section 5.2. Thus, the total SRAM capacity necessary for the 2 GB DRAM is 16 KB. According to the CACTI tool, for a 32 nm technology, the 16 KB SRAM requires 0.12 $\mathrm{mm}^2$ and consumes 3.6 mW standby leakage power. Because the typical chip size of a 2 Gb DRAM die is 40 $\mathrm{mm}^2$ and the logic die size is same as the DRAM die size, the area overhead of the registers becomes just 0.3 percent. Addiditionally, since the 2 GB DRAM device consumes 40 mW of refresh power [20], the 3.6 mW leakage power accounts for 9 percent of total refresh power.

## 5 EVALUATIONS

In this section, we evaluate the performance of EXTREME with full system simulations. First, we perform a simulation with assumptions of an infinite size of the page entry storage (PES) and immediate update of page bit map(PBM). After that, we explore suitable sizes for PES and the PBM update cycle(PUC) experimentally. Finally, we verify EXTREME with multiple program sequences, which have various sizes.

### 5.1 System Configuration

To evaluate EXTREME, we use gem5 full system simulator with the cycle-accurate DRAM simulator, DRAMSim2 [21], [22]. The details of the system configuration are listed in Table 2. In this work, we assume that the 3D-stacked DRAM device consists of eight layers of 2 Gb DRAM die, and has a capacity of 2 GB (see Fig. 13). Typical stacked DRAM dies operate with practical DDR3 timing constraints,

TABLE 2
Simulation Configuration

| | |
|---|---|
| CPU Type | ARM |
| CPU Frequency | 2GHz |
| DRAM Type | DDR3 |
| tAA/tRCD/tRP | 13.5 ns/13.5 ns/13.5 ns |
| DRAM Size | 2 GB (8 layers of 2 Gb die) |
| Rows | 32,768 |
| DRAM Frequency | 667 MHz |
| Number of Banks | 8 |
| Number of Ranks | 1 |
| Number of Columns | 8192 |
| Data Width | 64 bits |
| Row Buffer Policy | Open Page |
| Refresh Interval(tREF) | 64 ms |
| L1 Cache Size | 32 KB |
| L2 Cache Size | 1 MB |

and the logic die manages refresh operations of all DRAM dies in accordance with the EXTREME mechanism. In addition, the system is equipped with a single 3D-stacked DRAM device for simplicity, and its data width is 64 bits as in typical DRAM interfaces. Finally, we use PARSEC benchmark suites, which have various features in terms of data locality and memory footprint [23].

As mentioned in Section 1, high-capacity DRAM devices consume more refresh power than smaller ones. Therefore, evaluating EXTREME with a high-capacity DRAM device, such as 32 or 64 GB, can more effectively show the refresh power reduction. However, we evaluated EXTREME in this paper with a somewhat smaller 2 GB DRAM. There are several reasons why we evaluated the proposed scheme with a small-capacity DRAM, even though we noted in Section 1 that the refresh power is critical in high-capacity DRAMs. First, our proposed EXTREME scheme is deeply related to many parts of the system across OS, processor, cache and DRAM devices. Therefore, the evaluation requires a sophisticated simulator that can reflect the system in detail. In addition, to clearly evaluate the proposed scheme, it is more important to evaluate EXTREME using benchmark applications that accesses a larger space in a given DRAM, rather than just experimenting a higher-capacity DRAM. However,

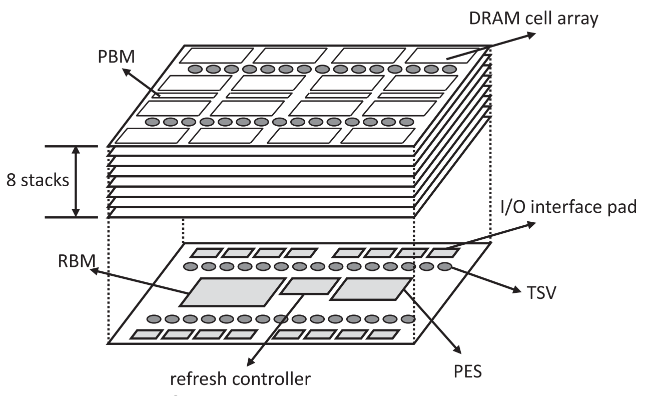
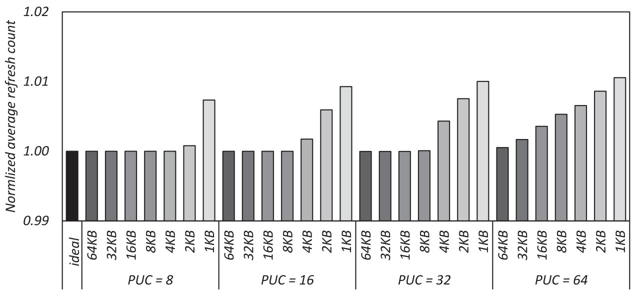
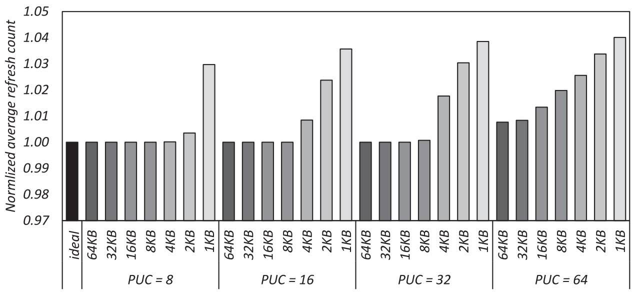


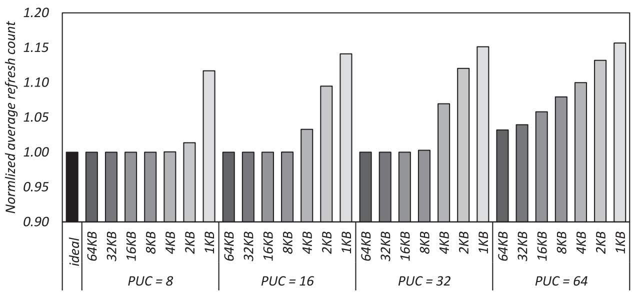
Fig. 13. An example of a 3D-Stacked DRAM device with EXTREME. The 3D-stacked DRAM device consists of multiple DRAM core dies and a single logic die, depending on memory configuration. The logic die has the SRAM registers, such as PES and RBM, and the control circuitries to manage the refresh operations.



(a) Single program



(b) Four programs



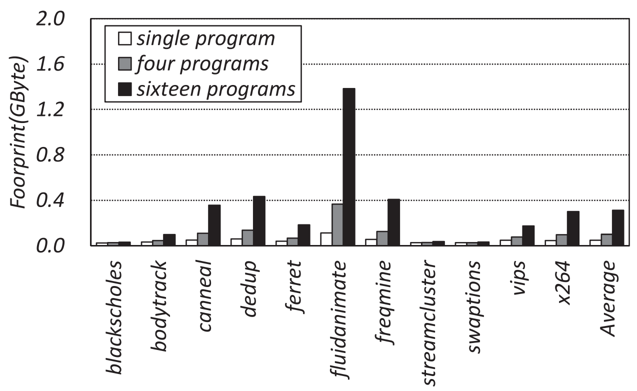
(c) Sixteen programs

Fig. 14. Normalized average refresh count of all applications. They are normalized for cases with an infinite-sized PES and an immediate PUC. As the size of the PES increases and the PBM updates more frequently, the refresh count converges to the ideal value. When 16 programs are running, the proposed DRAM device requires 8 KB PES and 32 PUC to get a refresh power similar to the ideal case.

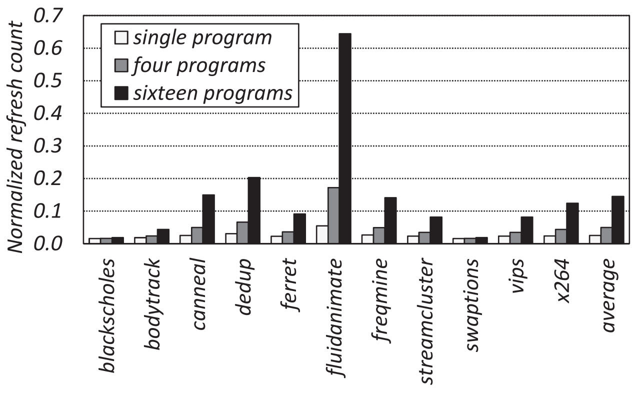
simultaneous testing of high capacity DRAMs and large applications is very limited in terms of implementation complexity and experiment time for the sophisticated full system simulator. Second, since the operation of EXTREME is deeply related to the size of the running applications rather than the DRAM capacity, the overhead caused by EXTREME is insignificant even if the capacity increases. Based on these considerations, we basically evaluated EXTREME with 2 GB capacity in this paper. However, we partially evaluate EXTREME using up to 8 GB of large DRAM in Section 5.5 to provide the effectiveness of the proposed scheme for the large DRAMs and discuss the performance and overhead.

## 5.2 Sensitivity Analysis of the Size of PES and PUC

It is important to determine the appropriate size of PES and the PUC during the design stage. To determine these parameters, we examine the refresh power reduction for various sizes of PES and the PUC through a full system simulation. Fig. 14 shows the normalized average number of refreshes in an ideal case with an infinite PES and the PUC



(a) Memory footprint of PARSEC application suites.



(b) Normalized average refresh count during the runtime of the applications.

Fig. 15. (a) Memory footprint of PARSEC applications (memory capacity = 2 GB). (b) Normalized refresh count with respect to the conventional DRAM during runtime of the applications. The refresh counts at runtime is proportional to the memory usage of the application.

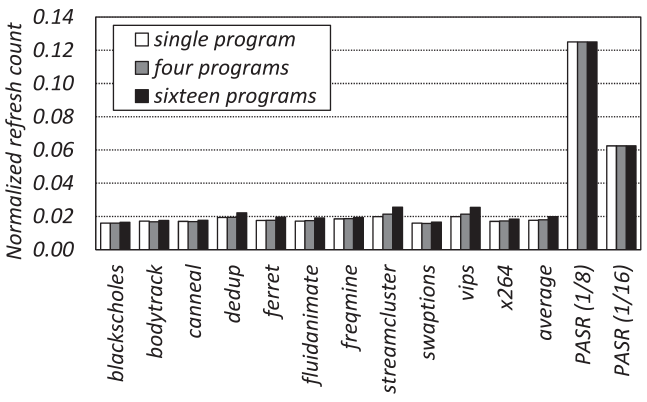


Fig. 16. Normalized refresh count with respect to typical DRAM after runtime of the applications (idle time). When an application is finished the OS deallocates the used pages and this deallocation is reflected into DRAM device and thus the refresh counts can be reduced. The graph shows the comparison with the PASR.

immediately updating. When a single program or four programs are running, even 1 KB of PES and 64 PUC can allow the refresh count of the scheme to be similar to that of ideal case, within bounds of 5 percent. In the case of sixteen programs, however, there needs to be at least 8 KB of PES to achieve over 95 percent of the refresh power reduction effect compared to the ideal case, when the PUC is maximized to 32. If the proposed scheme is designed to have 32 PUC, 3.125 percent of refresh peak power is added to the normal refresh cycle. Based on this analysis, we determine PES and the PUC to be 8 KB and 32, respectively.

## 5.3 Experiment Results

Memory systems are mainly in two time states. One is the program execution time and the other is the idle time after the program ends. At runtime, the proposed 3D-stacked DRAM device fetches the page information from the OS and refreshes only valid rows. Therefore, the refresh power becomes proportional to the size of the running program. On the other hand, the proposed DRAM device receiving the page free information at the end of the program resets the refresh bits of the RBM for the corresponding pages, and prevents the refresh operation of the deactivated pages. As a result, the refresh power is reduced to the value before the program started. In the following sections, we describe the experiment results of EXTREME by dividing them into runtime and idle time.

### 5.3.1 Runtime Experiments

The refresh power of the proposed 3D-stacked DRAM device is determined by the size of the running programs at runtime. To evaluate the refresh power reduction of the EXTREME scheme, we perform full system simulations with PARSEC benchmark suites. Fig. 15a shows the memory footprint of each application at runtime and Fig. 15b shows the normalized average refresh count of the proposed DRAM device. The figures indicate that the refresh count is increased or decreased in accordance with the size of the applications. Since computing systems generally do not fully utilize main memory, it provides a great opportunity to reduce refresh power. As a result, with the EXTREME technique, only 14.5 percent of the refresh power of a typical DRAM device can maintain all activated data, even when 16 programs are executed. Even though the refresh power can be changed according to the application size, it is obvious that EXTREME provides the most significant refresh power reduction effect. This is because the DRAM device with EXTREME has the information about activated data and its refresh is managed on row granularity rather than bank or sub-array.

### 5.3.2 Idle Time Experiments

When the system completes one or several applications, the OS releases the relevant pages. EXTREME exploits the deallocation information to reduce refresh power of the DRAM device. Fig. 16 shows the refresh count at idle time. The figure indicates that, even though an application such as fluidanimate consumes 64 percent refresh power at runtime, the count at idle time reduces to nearly 2 percent by the proposed scheme. We compare these results to the PASR, which is the most practical scheme, and show the refresh count to the right side of the graph. When the DRAM controller sets the DRAM to run with one-eighth array and one-sixteenth array PASR mode, the refresh count is decreased to 12.5 and 6.25 percent, respectively. The granularity of PASR's refresh management is greater than the granularity of EXTREME. This makes EXTREME better in terms of refresh power reduction.

## 5.4 Footprint Tracking Performance of EXTREME

To verify the footprint tracking performance of the proposed scheme, we measure the refresh count while

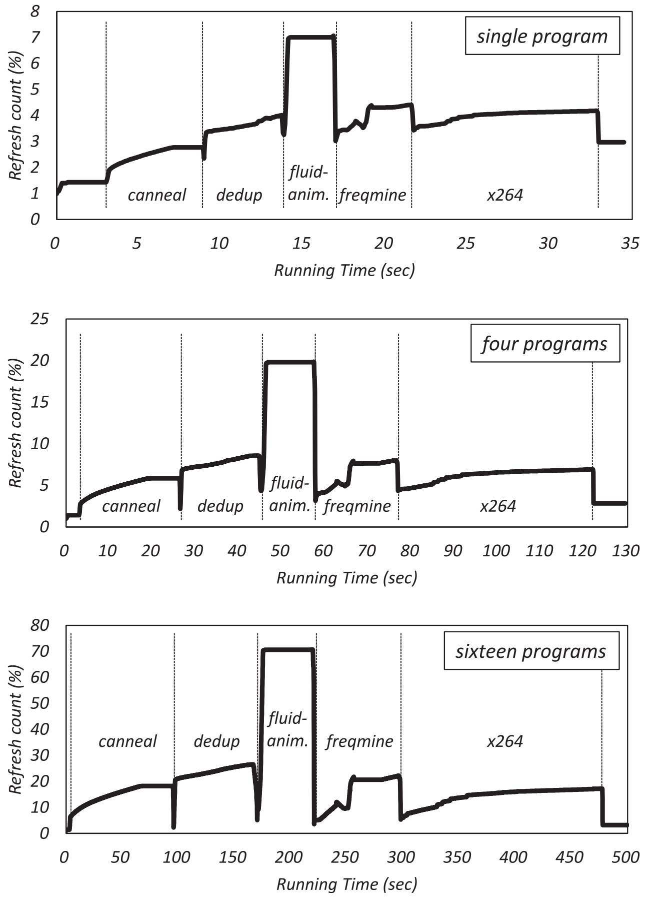


Fig. 17. Refresh count graph with execution time in various multiprogramming environments. As a single program or several programs are running, the number of refreshes increases as memory usage increases, and the number of refreshes decreases as the program completes.

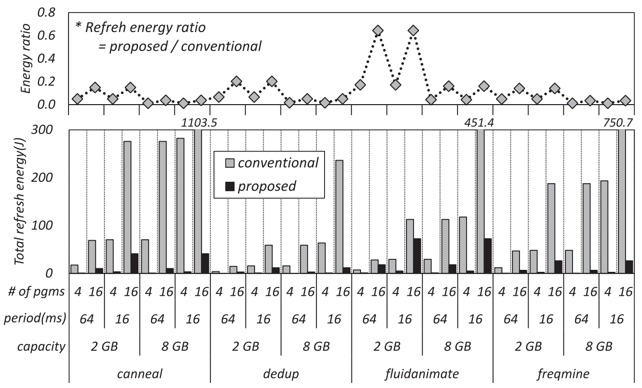


Fig. 18. Comparison of refresh energy according to the number of running programs, refresh period, and DRAM capacity of the conventional and proposed DRAM device. The upper figure shows the refresh energy ratio of the proposed DRAM versus the conventional one. The bottom figure shows the absolute amount of total refresh energy consumed during application operation.

sequentially running PARSEC benchmark applications that use memory space extensively. Fig. 17 shows the graph of refresh counts versus runtime for single and multiple programs. As the program runs, it starts to access memory, so the refresh counts increase proportionally. On the other hand, when the program ends, the OS frees the page frame, so the refresh counts decrease steeply according to the EXTREME mechanism. Since larger programs take up more memory space than smaller ones, the refresh counts increase with the size of the running programs. From the simulation results, we conclude that the proposed scheme normally follows the memory footprint of the programs, and the refresh power is reduced effectively.

### 5.5 Scalability Issues

This section explores various scalable parameters which can affect to the performance of EXTREME. The first parameter is the capacity of the DRAM device. As mentioned in Section 5.1, this paper basically evaluates EXTREME with a 2 GB 3D-stacked DRAM device. However, since our proposed scheme is more effective for the high capacity DRAMs, we additionally evaluate the refresh power reduction ability of EXTREME with an 8 GB DRAM and four benchmark applications which have a large memory access footprint. Fig. 18 shows the experimental results and compares the refresh energy between 2 and 8 GB. It shows that EXTREME reduces the refresh energy further on systems with larger DRAM capacity when the same application runs on a system with 2 and 8 GB memory capacity, respectively. This is because EXTREME's refresh power reduction ability is deeply related to the application size. That is, the refresh power of the conventional DRAM is proportional to the capacity of the DRAM, but the power of the EXTREME is proportional to the size of the application. Therefore, even if the capacity of the DRAM increases, EXTREME reduces more refresh power for a larger DRAM, if the size of the application is maintained.

The second parameter is the refresh period. 3D-stacked DRAM has a problem of heat dissipation. That is, the heat generated inside the chip can not be radiated to the outside, thereby degrading the retention time of the DRAM cells. This means that the refresh period must be reduced, and eventually it has a significant impact on the refresh power increase. In particular, this issue is more critical in the heterogeneous 3D-stacked DRAMs which contains a high-speed logic die. This paper sets the default period of the DRAM cells to 64 ms. However, in 3D-stacked DRAMs, its period can be reduced to 32, 16 ms or less. To evaluate the effect of EXTREME's refresh power reduction under the reduced refresh period conditions, we measured the refresh energy of the conventional DRAM and EXTREME. Fig. 18 shows the result. This result shows that as the refresh period decreases, the absolute amount of refresh energy that EXTREME decreases increases. It represent that the proposed scheme is more effective for the 3D-stacked DRAMs which uses reduced refresh period.

Though adopting EXTREME to the high capacity DRAMs is more effective in terms of refresh power reduction, it requires small overhead. First, as the DRAM capacity increases, the size of RBM should be expanded according to the Eq. (2). Second, as refresh period decreases, PBM update cycle can be reduced and it requires additional peak power management technology as mentioned in Section 4.3.2. However, since the time required to allocate and deallocate a page is very small compared to the total application execution time, it is a minor issue.

### 5.6 System Performance Analysis

EXTREME reduces the refresh power of the DRAM, but does not remove the refresh command itself issued by the

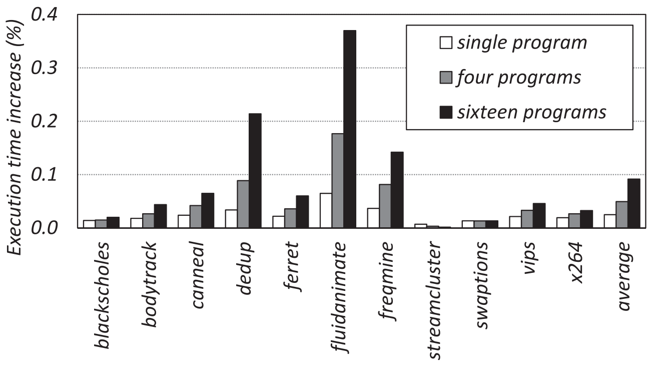


Fig. 19. Performance analysis. The page information generated by the processor is passed to the DRAM side through a cache flushing process, which is the only timing overhead of EXTREME.

DRAM controller. In other words, there is no timing performance improvement by the proposed scheme. This is because the EXTREME uses the latest CBR refresh policy, and the DRAM device is responsible for managing the refresh operation. Rather, the proposed technique requires some timing overhead. As we mentioned in Section 4.1, OS-generated pages must be sent to the DRAM side by flushing the caches of the processor. This cache flush takes some time and is the only timing overhead by EXTREME. However, the percentage that it occupies in the overall application runtime is very low. We measured the timing overhead due to the cache flushing, and Fig. 19 shows the result. It shows that when EXTREME runs with 16 programs, the execution time increases by an average of 0.09 percent. In some programs such as *fluidanimate* and *dedup*, it increases up to 0.38 and 0.21 percent respectively, but this is also a very small part of the overall run time.

## 6 Conclusion and Future Work

In order to reduce the refresh power of DRAM devices, we propose EXTREME, which is highly compatible with existing computing systems. The key idea is to design a DRAM device that pins the OS's page table in a specific physical address space and leverage it to manage its own refresh operations. Because the page table has information about valid pages, EXTREME can selectively refresh rows with valid data.

Pinning the page table to a specific memory space is a challenge to implement because it requires a lot of modifications to the OS or system. To overcome this, we split the page table management structure into two, and we use one part for the page directory and the other for the page table. By inspecting the page frame numbers(PFNs) of allocated or deallocated pages and putting them each into a structure, the page table can be pointed to a specific physical memory address. This method is highly compatible with current computing systems because it only modifies the OS without modification to the system or DRAM controller.

The DRAM industry is quite sensitive to cost, so, although EXTREME can effectively reduce refresh power, hardware design minimizing the refresh control logic is essential. To achieve this small logic, we propose a DRAM based register architecture, page bit map(PBM), and SRAM-based temporary storage page entry storage(PES) and fre-fresh bit map(RBM).

Through a full system simulation, the average refresh count of the 3D-stacked DRAM device with the EXTREME scheme and 2 GB capacity is reduced to 14.5 percent at runtime and 1.8 percent after 16 programs are completed. In real computing systems, several background applications are running and account for substantial memory space. Therefore, the decrement of the refresh count can be different from the simulation results. Since we implement the proposed scheme with row granularity, however, the refresh power efficiency can be maximized, unlike PASR which has fixed array granularity.

We expect the EXTREME scheme to be an ultimate solution for the low power memory systems of the extremely high capacity and high performance DRAM era. In addition, we plan to offload the management codes as well as the page table itself. We leave these components for future work.

## Acknowledgments

This work was supported in part by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (2016R1A2B4011799), by the ICT R&D program of MSIP/IITP. [2016(R7177-16-0233), Developement of Application Program Optimization Tools for High Performance Computing Systems], and by Samsung Electronics Company, Ltd., Hwasung, Korea.

## References

[1] Y. Joshi and P. Kumar, *Energy Efficient Thermal Management of Data Centers*. Berlin, Germany: Springer, Mar. 2012.

[2] J. Liu, B. Jaiyen, R. Veras, and O. Mutlu, "RAIDR: Retention-aware intelligent DRAM refresh," in *Proc. 39th Annu. Int. Comput. Archit.*, Jun. 2012, pp. 1–12.

[3] JEDEC, "JESD235: High bandwidth memory (HBM) DRAM," (2013). [Online]. Available: https://www.jedec.org/system/files/docs/JESD235.pdf

[4] Micron Technology, "Hybrid memory cube specification 2.0," (2014). [Online]. Available: http://www.hybridmemorycube.org/files/SiteDownloads/HMC-30GVSR_HMCC_Specification_Rev2.1_20151105.pdf

[5] K. Sohn, et al., "A 1.2 V 20 nm 307 GB/s HBM DRAM with at-speed wafer-level IO test scheme and adaptive refresh considering temperature distribution," *IEEE J. Solid-State Circuits*, vol. 52, no. 1, pp. 250–260, Jan. 2017.

[6] Samsung Electronics, "DDR3 SDRAM datasheet," (2006). [Online]. Available: http://www.samsung.com/semiconductor/products/dram/pc-dram

[7] Samsung Electronics, "DDR4 SDRAM datasheet," (2014). [Online]. Available: http://www.samsung.com/semiconductor/products/dram/pc-dram

[8] A. Sloss, D. Symes, and C. Wright, *ARM System Developer's Guide: Designing and Optimizing System Software*. San Mateo, CA, USA: Morgan Kaufmann, May 2004.

[9] A. Silberschatz, P. B. Galvin, and G. Gagne, *Operating System Concepts*. Hoboken, NJ, USA: Wiley, 2010.

[10] M. Ghosh and H. H. S. 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.

[11] T. Ohsawa, K. Kai, and K. Murakami, "Optimizing the DRAM refresh count for merged DRAM/logic LSIs," in *Proc. Int. Symp. Low Power Electron. Des.*, Aug. 1998, pp. 82–87.

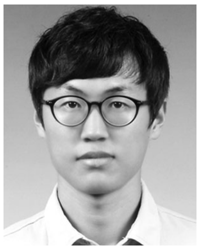
[12] R. K. Venkatesan, S. Herr, and E. Rotenberg, "Retention-aware placement in DRAM (RAPID): Software methods for quasi-non-volatile DRAM," in *Proc. 12th Int. Symp. High-Perform. Comput. Archit.*, Feb. 2006, pp. 155–165.

[13] S. Liu, K. Pattabiraman, T. Moscibroda, and B. G. Zorn, "Flikker: Saving DRAM refresh-power through critical data partitioning," in *Proc. 16th Int. Conf. Archit. Support Program. Languages Operating Syst.*, 2011, pp. 213–224. [Online]. Available: http://doi.acm.org/10.1145/1950365.1950391

[14] S. Baek, S. Cho, and R. Melhem, "Refresh now and then," *IEEE Trans. Comput.*, vol. 63, no. 12, pp. 3114–3126, Dec. 2014.
[15] 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*, Dec. 2009, pp. 337–346.
[16] Micron Tecnology, "Low-power function of mobile RAM-partial array self refresh (PASR)," Elpida Memory, Inc., Tokyo, Japan, Tech. Rep. E0597E10, 2005.
[17] J. Liu, B. Jaiyen, Y. Kim, C. Wilkerson, and O. Mutlu, "An experimental study of data retention behavior in modern DRAM devices: Implications for retention time profiling mechanisms," in *Proc. 40th Annu. Int. Symp. Comput. Archit.*, 2013, pp. 60–71. [Online]. Available: http://doi.acm.org/10.1145/2485922.2485928
[18] K. Kim and J. Lee, "A new investigation of data retention time in truly nanoscaled DRAMs," *IEEE Electron Device Lett.*, vol. 30, no. 8, pp. 846–848, Aug. 2009.
[19] H. Laboratories, "CACTI 6.0: A tool to model large caches," Tech. Rep. HPL-2009–85, 2009. [Online]. Available: http://www.hpl.hp.com/techreports/2009/HPL-2009–85.html
[20] Micron Technology, "Calculating memory system power for DDR3," Tech. Rep. TN-41–01, 2007. [Online]. Available: https://www.micron.com/resource-details/3465e69a-3616-4a69-b24d-ae459b295aae
[21] N. Binkert, et al., "The Gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, no. 2, pp. 1–7, Aug. 2011. [Online]. Available: http://doi.acm.org/10.1145/2024716.2024718
[22] P. Rosenfeld, E. Cooper-Balis, and B. Jacob, "DRAMSim2: A cycle accurate memory system simulator," *IEEE Comput. Archit. Lett.*, vol. 10, no. 1, pp. 16–19, Jan.-Jun. 2011.
[23] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The PARSEC benchmark suite: Characterization and architectural implications," in *Proc. 17th Int. Parallel Archit. Compilation Techn.*, 2008, pp. 72–81. [Online]. Available: http://doi.acm.org/10.1145/1454115.1454128

**Ho Hyun Shin** received the BS and MS degrees in electrical and electronic engineering from Korea University, Seoul, Korea, in 2003 and 2008, respectively. He has been worked as a senior engineer in the DRAM Design Team, Memory Division, Samsung Electronics. He is also currently working toward the PhD degree in the School of Electrical and Electronic Engineering, Yonsei University. His research interests include high performance DRAM design and system architecture. He is a member of the IEEE.

**Young Min Park** received the BS degree in electrical and electronic engineering from Yonsei University, Seoul, Korea, in 2015, where he is currently working toward the PhD degree in electrical and electronic engineering. His research interests include high performance computing system and CAD flow on near threshold voltage.


**Duheon Choi** received the BS degree in electrical and electronic engineering from Yonsei University, Seoul, Korea, in 2015, where he is currently working toward the PhD degree in electrical and electronic engineering. His research interests include memory architecture and system-level design.

**Byoung Jin Kim** received the BS degree in electrical and electronic engineering from Yonsei University, Seoul, Korea, in 2016, where he is currently working toward the PhD degree in electrical and electronic engineering. His research interests include memory architecture and system-level design.

**Dae-Hyung Cho** received the BSc and MSc degrees in electronic engineering from Hanyang University, in 1984 and 1988, respectively, and the PhD degree in electrical and computer engineering from the University of Illinois at Urbana-Champaign (UIUC), in 1995. From 1984 to 1991, he worked as assistant manager at Samsung Electronics, and worked as senior manager at Hyundai Electronics from 1995 to 1997. In 1997, he joined Intel Corporation at Santa Clara, California and worked at Intel as senior staff and TCAD Lab manager until 2002. In 2002, he re-joined Samsung Electronics as Sr. director where he managed high speed devices technology development for mobile applications until 2005. He has been serving as the senior advisor of EPFL (Swiss Federal Institute of Technology at Lausanne) since 2012. He also served in an individual consultant, executive advisor, and representative of worldwide IT companies in USA and Europe. Previously, he was founder and CEO of Ubeacon Technology which had developed WUSB/UWB chipset, where he was responsible for fund raising and strategic business development. His career spans 30 years of experience in the semiconductor industries for technology development and the global technology marketing area, where he has proven his enormous leverage in linking global technology and business network and relationship to build new IT technology enabled business.

**Eui-Young Chung** received the BS and MS degrees in electronics and computer engineering from Korea University, Seoul, Korea, in 1988 and 1990, respectively, and the PhD degree in electrical engineering from Stanford University, Stanford, California, in 2002. From 1990 to 2005, he was a principal engineer with SoC R&D Center, Samsung Electronics, Yongin, Korea. Currently, he is a professor in the School of Electrical and Electronics Engineering, Yonsei University, Seoul, Korea. His research interests include system architecture, bio-computing, and VLSI design, including all aspects of computer-aided design with the special emphasis on low-power applications, and flash memory applications. He is a member of the IEEE.