Partial Row Activation for Low-Power DRAM System

Yebin Lee[1] Hyeonggyu Kim[1] Seokin Hong[2] Soontae Kim[1]

[1]School of Computing, KAIST
{yblee87, hyeonggyu, kims}@kaist.ac.kr

[2]Samsung Electronics
seokin1.hong@samsung.com

ABSTRACT

Owing to increasing demand of faster and larger DRAM system, the DRAM system accounts for a large portion of the total power consumption of computing systems. As memory traffic and DRAM bandwidth grow, the row activation and I/O power consumptions are becoming major contributors to total DRAM power consumption. Thus, reducing row activation and I/O power consumptions has big potential for improving the power and energy efficiency of the computing systems. To this end, we propose a partial row activation scheme for memory writes, in which DRAM is re-architected to mitigate row overfetching problem of modern DRAMs and to reduce row activation power consumption. In addition, accompanying I/O power consumption in memory writes is also reduced by transferring only a part of cache line data that must be written to partially opened rows. In our proposed scheme, partial rows ranging from a one-eighth row to a full row can be activated to minimize row activation granularity for memory writes and the full bandwidth of the conventional DRAM can be maintained for memory reads. Our partial row activation scheme is shown to reduce total DRAM power consumption by up to 32% and 23% on average, which outperforms previously proposed schemes in DRAM power saving with almost no performance loss.

1. INTRODUCTION

Because of the growing demand for high-speed and large-capacity memory in today's computing systems, especially, servers and datacenters, DRAM accounts for a major portion of their total power and energy consumptions. Recent studies demonstrate that the memory system accounts for 25%–57% of the total power consumption of a system [1, 2, 3, 4]. Therefore, improving the power and energy efficiencies of the DRAM system is the key consideration in the memory architecture design. With the generalization of multi-core processors, multi-threaded and multi-programmed workloads have become prevalent in modern computing systems. Following this trend, memory traffic has become heavier and more random, which implies that more DRAM row activations will be required owing to intermixed memory requests from workloads. Moreover, an entire row should be activated even though only a small portion of the row data is accessed; this problem is called *row overfetching*. Owing to this inefficiency, row activations account for a major portion of the DRAM power and energy consumptions. From our observation, row activation and bank precharge pairs account for up to 33% and average 25% of total DRAM power consumption. In addition, DRAM I/O power consumption is another major contributor to the DRAM power and energy consumptions in modern DRAM systems [3, 5, 6, 7]. As memory traffic and DRAM bandwidth increase, large amount of power/energy is consumed by DRAM I/O. According to our observation, DRAM I/O power consumption including read I/O, write on-die termination (ODT), and read/write termination accounts for up to 19% and average 14% of total DRAM power consumption.

To reduce row activation power consumption, recent studies have addressed the row overfetching problem. Fine-grained activation [8, 9] is one of the schemes that activate a small portion of a DRAM row for reducing row activation power consumption. Although the idea of fine-grained activation is interesting, it incurs severe performance degradation because the n-bit prefetching mechanism, which is the key to modern high-speed DRAM systems, is inevitably broken down. In contrast, Half-DRAM [10], which is a clever design that vertically divides DRAM matrices of cells (MATs) into halves, has been proposed to solve performance problem of previous fine-grained activation schemes while achieving half row activation. As we will see later in this paper, there is significant room for power saving beyond half row activation. However, the sophisticated design is crucial, which has low hardware overhead and design complexity. From this viewpoint, extending Half-DRAM design to support finer granularity row activation than half would not be suitable because large overheads are inevitably involved when it is extended to one-fourth or one-eighth row activation. In this work, we propose a partial DRAM row activation scheme to address the design challenges of prior schemes and to achieve further power saving with finer granularity row activation and reduced DRAM I/O switching activity.

In our proposed scheme, a partial row can be activated on a memory write request to minimize row activation power consumption and a full row is activated on a memory read request to maintain the full bandwidth that DRAM can provide. This approach is inspired by the following observations. First, the row buffer locality of the write traffic is poor; row activations incurred by write requests account for a large proportion of the total row activations because they do not show good reuse behavior. Second, not all words in a cache line are actually dirty; clean words do not need to be written to DRAM, and, therefore, a corresponding portion of a row to the clean words does not need to be activated and

2378-203X/17 $31.00 © 2017 IEEE
DOI 10.1109/HPCA.2017.35

write I/O power consumption can be reduced as well. Third, the existence of fine-grained structures in DRAM provides an opportunity to mitigate the row overfetching problem; the internal architecture of DRAM is composed of hierarchical fine-grained structures (i.e., MATs and sub-arrays). To support the partial row activation architecture, the cache memory hierarchy is extended to provide dirtiness information at the word level granularity and small hardware addition and modification are made to DRAM chip.

Our paper makes the following key contributions:

- We identify that the row overfetching problem is more severe for writes than reads, and there is significant asymmetry between reads and writes in terms of row buffer locality.
- We revisit the phenomenon showing that not all words of a cache line are dirty to mitigate intra-chip row overfetching.
- We propose an effective approach that reduces DRAM row activation and write I/O power consumptions by activating partial rows ranging from a one-eighth row to a full row and transmitting a part of cache line data that must be written to DRAM.
- We quantify the impact of our partial row activation architecture on the performance, DRAM power/energy consumption and energy-delay product compared to the previously proposed schemes.

2. BACKGROUND AND MOTIVATION

2.1 DRAM Preliminary

2.1.1 DRAM Organization and Structure

A commodity DRAM system is designed as a hierarchical organization and structure as shown in Figure 1. A channel is composed of a memory controller and contains one or more ranks that share the address/command and data buses. Because ranks share the channel, only one rank can be selected and used at a time. A rank consists of multiple DRAM chips physically organized as a form of memory module called dual in-line memory module (DIMM) and all chips in a rank operate in lockstep. Because the I/O data width of a single chip is limited (i.e., 4-, 8-, or 16-bit), multiple chips are populated in a rank to form a wide data bus (i.e., 64-bit). In this paper, we use 2Gb x8 DDR3-1600 chip [11], unless stated otherwise. Inside a chip, 8 banks are deployed as DRAM cell arrays. Banks can be accessed independently, and thus, bank-level parallelism is extensively exploited to assure memory bandwidth. Similarly, there are channel-level and rank-level parallelisms available in the DRAM system, wherein channels are completely independent whereas ranks have limited parallelism due to the rank-to-rank switching time. Subsequently, a bank is comprised of 64 sub-arrays each of which consists of 16 MATs. Because implementing a large bank results in unreliable sensing of bitlines and long latency of activation/precharge, a bank is divided into an array of tiles (sub-arrays and MATs) [12, 13, 14]. Note

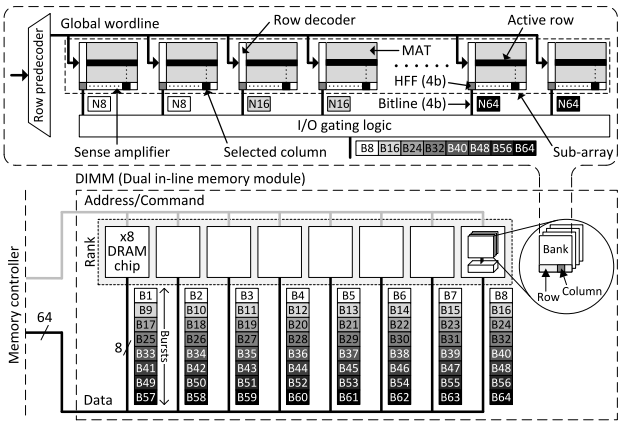


Figure 1: Conventional DRAM system organization, structure, and data mapping. B*x* is a byte of a 64B cache line and N*x* is a nibble of byte B*x*.

that only one sub-array is shown in Figure 1 for simplicity of presentation. A MAT is formed as a 512 $\times$ 512 matrix of DRAM cells. Each MAT in a sub-array has a local row decoder, a sense amplifier, and a helper flip-flop (HFF). Once a row address is fed into the global row decoder (row predecoder), a sub-array is selected and the row address is delivered to the local row decoders of MATs in the sub-array through the global wordline. Then, all the local row decoders drive their target rows, which are sensed and amplified by their sense amplifiers (row buffers). The data of the selected columns are then transferred through HFFs and I/O gating logic to/from the read or write FIFOs. A HFF is responsible for latching the selected data and relaying them on the internal data bus [12]. Finally, the data can be read from the FIFO to the data bus in a pipelined manner and burst fashion. This is enabled by the *n*-bit prefetch architecture (where *n* is 2 in DDR, 4 in DDR2, and 8 in DDR3 and the prefetch width is *n* $\times$ data width of chip), which is the key to modern high-speed DRAM technologies.

2.1.2 Data Mapping

As shown in Figure 1, a 64-byte cache line is delivered throughout 8 bursts to/from the 64-bit data bus of the DRAM system. At each burst, an 8-byte word of the cache line is moved to/from all chips, each of which receives or sends 1-byte data (B*x*) of the word. Thus, each byte of words in a cache line is scattered into each chip, which can support the "critical word first" capability of the modern cache architecture. Within a chip, each byte is divided into two nibbles (N*x*) and each nibble is linearly mapped to each MAT in a sub-array because each MAT is connected to four global bitlines. Thus, each group of two MATs contains a byte data of each word in a cache line. Because of this characteristic of data mapping, prior fine-grained activation schemes inevitably necessitate modifications to data mapping for delivering a 64B cache line to/from partially selected MATs. In case of selecting two MATs for fine-grained activation, the entire cache line must be mapped within the two MATs, which results in 64 bursts for transferring a 64B cache line. In other words, the bandwidth is reduced to one-eighth,

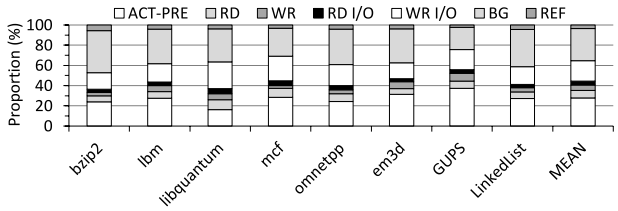


Figure 2: DRAM power consumption breakdown. ACT-PRE, RD, WR, RD I/O, WR I/O, BG, and REF indicate a pair of row activation and bank precharge, read, write, read I/O, write I/O, background (standby), and refresh power consumptions, respectively.

Table 1: Memory characteristics.

Benchmark	Row buffer hit rate (%)		Memory traffic (%)		Row activation (%)	
	Read	Write	Read	Write	Read	Write
bzip2	32	1	69	31	60	40
lbm	29	18	57	43	54	46
libquantum	73	48	66	34	50	50
mcf	18	1	79	21	76	24
omnetpp	47	2	71	29	57	43
em3d	5	1	51	49	50	50
GUPS	3	1	53	47	52	48
LinkedList	4	1	65	35	64	36
average	26	9	64	36	58	42

which incurs significant performance degradation. Therefore, a sophisticated design is essential when implementing a fine-grained activation technique.

2.2 Motivations

2.2.1 Row Overfetching Problem

As we have seen in Section 2.1, an entire row has to be activated even though only a small portion of the row data is accessed. In our baseline, an 8K-bit row is activated for transferring 64-bit data to/from a chip. From a rank-level point of view, an 8KB row is opened even though only a 64B cache line needs to be accessed per request. Figure 2 shows the DRAM power consumption breakdown, in which we can see that significant DRAM power is consumed by row activation and bank precharge pair (ACT-PRE). See Section 5.1 for the detailed experimental setup (single-core is used to see raw impacts for the remaining motivational results). The row activation power is mainly consumed on the local bitlines and sense amplifiers, where activation and precharge operations manifest themselves. Moreover, the activation power is proportional to the number of bitlines being activated. This power inefficiency of row activation will increase in future DRAMs, which will have larger capacities and more bitlines [10]. Ideally, row overfetching is beneficial for performance because it avoids repeated activations if multiple requests access the same row (row buffer hit). In addition, it improves power efficiency because the activation power can be amortized over these accesses. Unfortunately, the application interference in modern CMP systems makes memory traffic more random, which lowers the opportunity of row buffer reuse [8, 15]. In addition, the memory controller itself limits the number of row buffer hits to prevent starvation and fairness problems among requests [15].

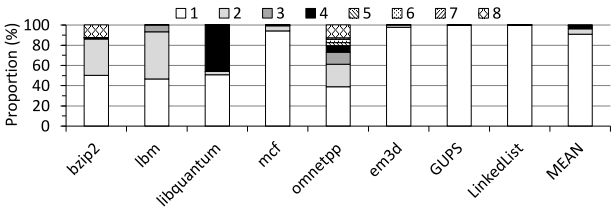


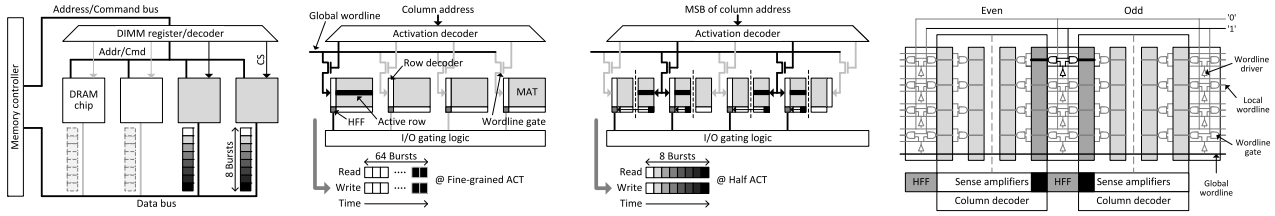
Figure 3: Proportion of dirty words in a cache line when the line is evicted from the LLC.

2.2.2 Locality Asymmetry of DRAM Read and Write Accesses

Through our experiments, we have found that there is significant locality asymmetry between DRAM read and write accesses. Table 1 shows various memory characteristics of benchmarks we used in this paper. The table shows the proportions of reads and writes for row buffer hit rates, memory traffic, and row activations. As shown in Table 1, write accesses show poor row locality, which indicates that they do not utilize the row buffers of DRAM well and write accesses result in frequent row activations owing to low row buffer hit rates. For example, in the *omnetpp* benchmark, only 2% of write accesses hit the row buffer (opened row) whereas half of the read accesses hit the opened row. Even though the proportion of write requests is 29% out of all memory requests, 43% of row activations are incurred by write accesses. Because read requests are critical to performance, modern memory controllers prioritize read requests over write requests along with first-ready first-come-first-served (FR-FCFS) request scheduling [16]. The performance impact of write requests can be somehow hidden but the activation power consumption cannot be avoided.

2.2.3 The Opportunity from Rethinking Fine-Grained Activation

As we previously discussed, row activations caused by write accesses are a major contributor to the total row activation power consumption. To address this issue, we recall the concept of the fine-grained activation approach as a dealmaker. In addition, an interesting phenomenon makes partial row activation feasible and leads to the core of this work; not all data in cache lines are dirty [17] and thus activating only a portion of a row that needs to be actually written can minimize write row activation power consumption. Figure 3 shows the proportion of dirty words in a cache line when the line is evicted from the LLC. Since the fine-grained activation technique requires modification of data mapping and incurs severe performance degradation, naively adopting it is too harmful [10]. However, rethinking fine-grained activation for write accesses can minimize power consumption of write row activations without significant loss of performance. As shown in Figure 3, there are not many dirty words in cache lines written to the DRAM system. Therefore, partially activating a row for those cache lines and selectively transmitting them can reduce significant row activation and write I/O power consumption.



(a) Skinflint DRAM system. (b) Fine-grained activation. (c) Half-DRAM. (d) Half-DRAM extension (1/4 row).

Figure 4: Comparison of skinflint DRAM system (SDS), fine-grained activation (FGA), Half-DRAM, and Half-DRAM extended to 1/4 row activation. For simplicity of presentation, we assume that one MAT can provide 8-bit data and bitlines are not shown in figure (c). CS in figure (a) indicates chip select.

3. RELATED WORK

Recent studies have focused on reducing DRAM row activation and I/O power consumptions which account significant portion of the DRAM power consumption in modern DRAM systems. In this section, we briefly discuss previously proposed schemes and qualitatively compare them with our partial row activation scheme.

Inter-chip row activation. A sub-rank memory system [4, 8, 18, 17, 19, 20, 21, 22, 23] is one of the approaches that target inter-chip row activation power reduction. In these schemes, conventional coarse-grained rank structure is divided into fine-grained ranks and thus fewer chips are accessed on memory requests. The state-of-the-art schemes, Dynamic granularity memory system (DGMS) [18] and Skinflint DRAM system (SDS) [17], have been proposed to overcome the latency problem of the sub-rank approach while also realizing the power and energy benefits of the sub-rank organization. In particular, the concept of SDS is close to our proposed scheme in the context of aiming write memory accesses for activation power reduction. Figure 4(a) shows SDS scheme. However, our proposed scheme differs from SDS in the following aspects. Our proposed scheme addresses intra-chip row activation whereas SDS targets inter-chip row activation. Because of inter-chip architecture, SDS can avoid chip accesses when all bytes at the same byte positions of all words in a cache line are clean by comparing old and new data. However, our scheme can adjust row activation granularity depending on the number of dirty words in a cache line. Thus, our scheme requires lighter support from the cache hierarchy and covers more write accesses to DRAM system. Specifically, our scheme reduces average row activation granularity by 42% whereas SDS can reduce average chip access granularity by only 16%, which shows better coverage and higher impact on the power saving with our proposed scheme.

Intra-chip row activation. Several work re-architecting internal structures of DRAM have been proposed to address the row overfetching problem of intra-chip row activation [8, 9, 10, 24, 25]. Among them, we generally denote fine-grained activation [9] and selective bitline activation [8] schemes as *fine-grained activation* (FGA) unless specifically stated otherwise. As discussed in Section 2.1, modern DRAM architectures employ small groups of DRAM cells (sub-arrays and MATs) to form a large bank due to latency, power, and reliability issues. The key idea of the fine-grained activation approach is to exploit the fine-grained characteristic of the DRAM structure for row activation power reduction. As shown in Figure 4(b), FGA selects one of the MATs in a sub-array on a row activation command using an extra hardware (activation decoder and wordline gate). Once a MAT is selected, then the local row is activated; a 64B cache line is fetched over 64 bursts (32 memory cycles) in this example. This is because n-bit prefetch capability of DRAM cannot be exploited anymore owing to a limited number of active HFF and MAT, and because the data mapping is changed to place the entire cache line within one MAT. In contrast, as shown in Figure 4(c), Half-DRAM [10] divides each MAT into halves to enable half row activation but exploits all HFFs cleverly even in inactive sub-MATs to support n-bit prefetch mechanism. Thus, full bandwidth of DRAM can be maintained while achieving half row activation and accompanying row activation power saving. In case of extending Half-DRAM as shown in Figure 4(d) to obtain further power saving in one-fourth or one-eighth row activation, the area overhead is expected to be 4x and 8x compared to our scheme (12% and 24% of DRAM die area [25]), which would not be suitable for finer granularity row activation. Unlike prior studies, our proposed scheme differentiates row activations into read and write cases. This asymmetric design can overcome bandwidth degradation of fine-grained activation by activating full rows for read requests and maintaining n-bit prefetch capability. In addition, large amount of row activation power can be reduced by activating partial rows for write requests. Moreover, significant write I/O power can be saved because only necessary data are written to DRAM in our scheme. We quantitatively evaluate the impact of FGA and Half-DRAM along with our PRA in Section 5.2.2 and we conduct a case study of using PRA with Half-DRAM in Section 5.2.3.

I/O energy consumption. DRAM I/O energy is consumed by I/O switching activity, which mainly dissipates energy on the data bus and I/O termination. Because modern DRAMs have adopted termination schemes to increase signal integrity for high-speed data transfer, I/O energy has become one of the dominant portion in DRAM energy consumption. Several work have been proposed to reduce DRAM I/O energy and power consumptions using data encoding schemes [7, 26, 27, 28]. These schemes focus on reducing hamming weight of the data. On the other hand, our proposed scheme simply reduces I/O switching by trans-

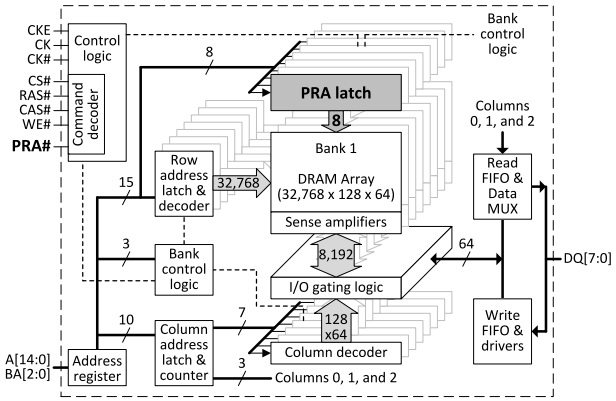


Figure 5: Overview of the partial row activation architecture. 2Gb x8 DDR3 DRAM chip [11] is shown.

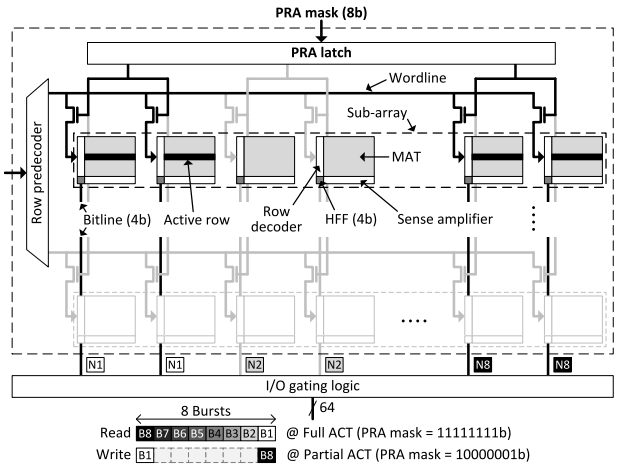


Figure 6: Detailed view of the partial row activation architecture in a bank. The bank has 64 sub-arrays and each sub-array contains 16 MATs.

mitting no data which do not need to be written to DRAM. Our scheme does not necessitate data encoding schemes.

DRAM-aware writeback. Several earlier schemes have been proposed to improve system performance and DRAM power efficiency by proactively writing back dirty cache lines mapped to the same DRAM row [29, 30, 31]. To this end, the last-level cache is queried and proactively writes back dirty cache lines of a DRAM row when any dirty line from that row is evicted from the cache. Although many row activations can be amortized over DRAM-aware write-backs, significant extra tag lookups are necessary to identify cache lines which will be written back to DRAM [29, 30]. In contrast, the Dirty-block index (DBI) [31] enables DRAM-aware writeback more efficiently because it separates dirty bits from the tag store to reduce significant tag lookup costs. These schemes focus on improving efficiency of row overfetching by increasing row buffer reuse. On the other hand, our scheme focuses on reducing the cost of row overfetching itself by minimizing row activation granularity. We explore a case study of our PRA combined with DBI in Section 5.2.3.

4. PARTIAL ROW ACTIVATION DESIGN

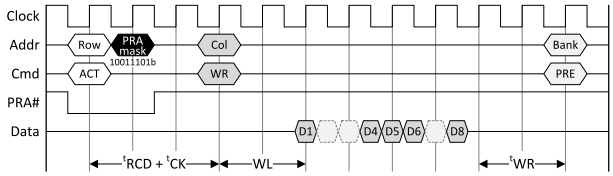
4.1 Partial Row Activation Architecture

4.1.1 Chip Internal

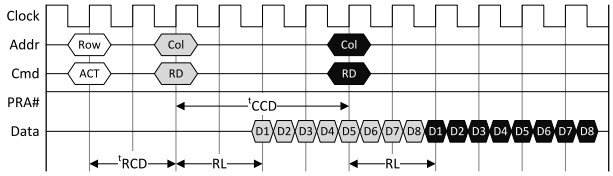
Figure 5 shows the overview of our proposed partial row activation (PRA) architecture in a DRAM chip. Unlike fine-grained activation techniques, multiple local rows in MATs can be selectively and simultaneously activated with PRA. This is because words of a cache line are mapped to different MATs in a sub-array. In this way, we can avoid repeated accesses, which are inevitably involved in prior fine-grained activation schemes. For example, if the first, second, and eighth words in a cache line are dirty, corresponding MATs and local rows mapped to these dirty words must be selected and activated for write access. To this end, we add a partial row activation (PRA) latch per bank and a PRA command to the command decoder of DRAM. In addition, the PRA mask needs to be delivered from the memory controller to the DRAM chips upon a row activation command. This can be implemented by exploiting the address bus of DRAM to avoid adding extra pins for PRA mask delivery. In our baseline DRAM chip, an 8-bit PRA mask is required for word-granularity writes and partial row activations. Such an implementation approach can also be found in [17]. A PRA latch is responsible for holding a PRA mask delivered from the memory controller via an address bus, and a PRA command pin is driven by the memory controller for indicating that a row activation is either PRA or a normal full row activation. We have several design alternatives and overhead issues when implementing PRA. A detailed discussion and analysis on implementation of PRA is given in Section 4.2.

4.1.2 Bank Internal

Figure 6 shows a detailed view of our PRA architecture in a bank. As shown in Figure 6, two MATs are grouped together and controlled by one bit of the PRA latch. This is because each MAT is connected to 4 bitlines and thus each group of two MATs need to be selected/unselected in lock-step to accommodate each byte of words in a cache line. Depending on the PRA mask, local rows in each group of two MATs can be activated or deactivated upon a row activation command. For example, if a PRA mask is 10000001b, the first and eighth groups of two MATs are selected and the row address from the global row decoder is relayed on the local row decoders of selected MATs through wordline gates. Then, local rows of the selected MATs are sensed and amplified (activation) by their sense amplifiers and particular columns of activated rows are projected to the I/O gating logic through their HFFs. Once a partial row is activated, the memory controller sends actual data of dirty words in a cache line via the data bus throughout 8 burst transactions. If the PRA mask is 11111111b, all MATs are selected and full row activation is performed. As explained in Section 2.1, each chip receives one byte of words in a cache line at each burst because the data width of our baseline chip is 8-bit. The first bytes of all words are mapped to the first chip, the second bytes of all words are mapped to the second chip,



(a) A case for partial row activation (delayed by ^{t}CK compared with conventional row activation timing).



(b) A case for full row activation (same as conventional row activation timing and applicable to write case, too).

Figure 7: Partial row activation timing.

and so on. Therefore, our design assures that all dirty words of a cache line can be stored without repeated accesses and modifications to the data mapping.

4.1.3 Timing

Figure 7 shows the PRA timing for the cases of partial row activation and normal full row activation. The timing parameters shown in Figure 7 are row activation to read/write delay (^{t}*RCD*), clock period (^{t}*CK*), write latency (*WL*), write recovery time (^{t}*WR*), delay between read/write (^{t}*CCD*), and read latency (*RL*). As shown in Figure 7(a), PRA can activate partial rows on write accesses for activation power reduction. A partial row activation (PRA) and normal full activation can be determined by pulling down or up the PRA command pin (PRA# in which '#' means active-low) alongside the row activation command (RAS# shown in Figure 5). In the case of PRA (PRA pin is pulled down), a row address is first buffered by a row address latch but a chip does not perform row activation immediately. Instead, it waits for a PRA mask, which will be delivered through the address bus in the next cycle. Right after the row activation command, the PRA mask is transferred to a PRA latch on the target bank. After the PRA mask is available in the PRA latch, a partial row can be activated. Once the partial row is activated, the memory controller can issue a write command with a column address (after ^{t}*RCD*+^{t}*CK*) and actual data can then be transferred after *WL*. As shown in Figure 5, byte data are sequentially fed into the write FIFO in a chip at each burst through the data bus (DQ) and the data in the FIFO are overwritten to the sense amplifiers of selected MATs. Because only a selected partial row is activated, the data mapped to unselected MATs are ignored (treated as "don't care"). A bank can be precharged finally after ^{t}*WR* from the end of burst transactions. For the case of full row activation as shown in Figure 7(b), the PRA pin is pulled up with the row activation command. This instructs a chip to activate a full row immediately. Thus, the column read command can be issued after ^{t}*RCD*. Once the full row is activated, the data are fetched from a chip throughout 8 bursts. Because PRA can maintain the full row open, consecutive accesses heading to the opened row can hit the row buffer, which results in fast accesses and activation power reduction by removing repeated row activations.

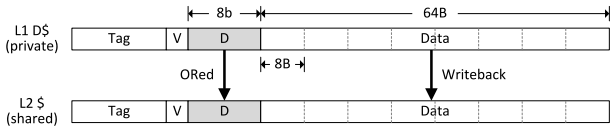


Figure 8: Fine-grained dirty bit (FGD) in cache hierarchy for partial row activation. V indicates a valid bit and D means dirty bits.

In commodity DRAM, two timing parameters are defined to prevent frequent row activations from generating huge power consumption that exceeds a predefined threshold of peak power consumption. ^{t}*RRD* is a delay between two consecutive row activations in different banks and ^{t}*FAW* is four activation window that limits row activations to a maximum of four times during a given time frame. These timing parameters restrict row activation frequency and thus diminish the opportunity of potential performance gain from better memory parallelism. By activating partial rows, we can take advantage of relaxed ^{t}*RRD* and ^{t}*FAW* timing constraints. We can issue more and faster write requests, which cannot be carried out in the conventional DRAM system due to timing and power constraints. Because partial row activations are conducted on write requests and write accesses are performed in background, the performance gain of relaxed ^{t}*RRD* and ^{t}*FAW* may not be significant in general cases. However, we can clearly have benefits from relaxed timing constraints, which will be larger in case applications generate write-intensive memory traffic and restricted close-page row buffer management policy is used. Under heavy write traffic, read requests can be delayed because write requests are drained frequently to prevent queue overflow. In case of restricted close-page policy, every read and write request involves row activation and bank precharge operations in which ^{t}*RRD* and ^{t}*FAW* timing constraints significantly limit the bandwidth of DRAM.

4.1.4 Cache Hierarchy Support

In PRA, only dirty words in a cache line are written to DRAM. To support this, fine-grained dirty bits (FGD) in the cache hierarchy are required. Figure 8 shows FGD on the cache hierarchy to support PRA. A FGD is similar to a sector cache [32], which divides a cache line into multiple sectors. In FGD, the 64B data field of a cache line is logically divided into eight 8B-word segments and thus 8 dirty bits are needed in a cache line. The additional cost is the storage for extra dirty bits (extra 7 bits per cache line when a 64B cache line is divided into eight 8B logical segments). Each of 8 dirty bits indicates whether a corresponding word is dirty or clean. With store instructions, a cache line in the L1 data cache is written and its dirty bits are updated accordingly. When a dirty cache line is evicted from the L1 cache, its dirty bits are ORed with the dirty bits of the corresponding cache line in the L2 cache. When a dirty cache line in the L2 cache is evicted, its corresponding dirty bits are delivered to

the memory controller alongside the cache line data. The delivered dirty bits are then used as a PRA mask to activate the partial row mapped to the dirty words of the evicted cache line.

4.2 Overhead Analysis

In PRA, an extra pin is required for the PRA command in a DRAM chip. In our baseline DRAM system, there are unused pins available in a chip and DIMM, which can be used as the PRA command (generally, 4–6 pins and 1 pin are unused in DDR3 [11] and DDR4 [33] depending on the chip package and form-factor of DIMM). Also, we can use existing pins that are not used during row activation instead of using extra pin. This multi-functionality of a pin is already realized in commodity chips and DIMMs (e.g., A10/AP pin in DDR3, WE/A14 pin in DDR4, etc.). Alternatively, PRA command can be implemented by exploiting data mask (DM) feature of conventional DRAM architecture. With this design, PRA mask can be delivered via DM pin along with the write burst prior to row activation. However, this approach has following limitations and drawbacks. DM pin is sometimes used as TDQS (termination data strobe) to preserve signal integrity when the different DIMM configurations are mixed within the same DRAM system. In this case, we cannot exploit DM pin for PRA command. Subsequently, this approach potentially limits rank/bank parallelism of DRAM system because the write buffer of a chip is occupied until ongoing partial row activation is complete.

To deal with partial row buffer misses of PRA, the information of a partially activated row per bank needs to be maintained in the memory controller. A small storage is required, which is only 64 bits per rank (an 8-bit PRA mask for each of 8 banks in our baseline chip). Because the conventional memory controller strictly tracks all timing constraints and finite-state machine (FSM) of DRAM, tracking PRA information induces only small complexity in the memory controller.

We estimate energy and area overheads of FGD in the cache hierarchy using CACTI-3DD [34] at the 22nm technology node. Extra 7 bits need to be added to the existing one dirty bit. For a 32KB L1 cache, area, per-access dynamic energy, and leakage power overheads are estimated to be 0.31%, 0.12%, and 1.26%, respectively. For a 4MB L2 cache, those numbers are 1.09%, 0.41%, and 1.39%, respectively. Because accessing the off-chip main memory can cost 250× more energy consumption than accessing an on-chip cache [35, 7], overheads caused by FGD are very small compared with the DRAM system.

In the PRA architecture, a PRA latch is added to each bank in a chip. As a result, eight PRA latches per chip are required, where each latch stores an 8-bit PRA mask. To estimate the overhead of the PRA latches, we leverage the approach used in [14]. Scaling the area of a previously proposed latch design [36] to the 20nm process technology, each PRA latch has an area of 1.97 μm^2. Overall, eight 8-bit PRA latches incur a 0.13% area overhead compared to a 2Gb DRAM chip (11.9mm^2 die area as shown in Table 2). Similarly, normalizing the latch power consumption at the 20nm technology node, a PRA latch consumes 3.8 µW power for

Table 2: DRAM die area and row activation energy breakdown of a 2Gb x8 DDR3-1600 DRAM chip.

Area (mm^2)			
DRAM cell	4.677	Sense amplifier	1.909
Row predecoder	0.067	Local wordline driver	1.617
Total area (including other components)			11.884
Energy per MAT (pJ)			
Local bitline	15.583	Local wordline	0.046
Local sense amplifier	1.257	Row decoder	0.035
Total row activation energy per MAT			16.921
Energy per bank (pJ)			
Row activation bus	17.944	Row predecoder	0.072
Total row activation energy per bank			288.752

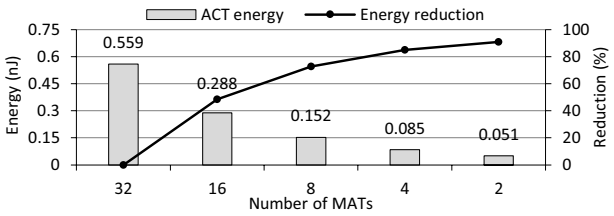


Figure 9: Row activation energy consumption depending on the number of MATs activated.

each row activation. This is a 0.017% power overhead compared to the power consumption of row activation.

As fine-grained activation does, PRA needs extra gates to control the local wordlines of MATs to be asserted or not asserted and thereby to implement partial row activation. Due to the capacitance and resistance problems of a long metal wordline strap (global wordline), modern DRAM architecture leverages hierarchical wordlines that are divided into multiple sub-wordlines stitched to the polysilicon wordlines (local wordlines), each of which has a dedicated driver and spans a single MAT throughout a local row decoder [12, 13, 37]. To control a group of two MATs to be selected or deselected (the minimum activation granularity in PRA), AND gates need to be deployed [24] for 512 local wordlines of two MATs and connected to a PRA latch in a bank though the existing metal layer where global bitlines and column select lines are placed. According to the practical analysis in [25], the area overhead due to the AND gates is estimated to be about 3% in our baseline 2Gb DRAM chip [11] and wire overheads are trivial because a single wire is necessary per MAT (16 wires per bank) between a PRA latch and wordline gates. There might be several design choices in implementation. However, none of them will show more overhead because our estimation is based on the worst case option.

Because errors in DRAM are common in modern computing clusters and failures are costly [38], server-class memory systems adopt error protection techniques such as error-checking and correcting (ECC) codes [39] and chipkill-correct [40]. Since PRA does not change the data mapping of the DRAM system, existing error protection strategies that can be applied to the conventional DRAM system can also be used in PRA. For example, in x72 ECC protected DIMMs [41, 42], making a PRA command pin of a particular DRAM

chip that stores ECC codes always pulled up (e.g., connected to VDD on the PCB) will activate a full row even for write requests all the time. Thus, a chip for ECC codes can receive and send the ECC data to/from the full row while remaining 8 chips for cache line data can benefit from partial row activations. Because a PRA command pin is not asserted on a ECC chip, a PRA mask on the address bus is not treated as valid in the ECC chip.

5. EXPERIMENTAL RESULTS

5.1 Methodology

5.1.1 DRAM Power Model

To establish the power model of partial row activation, we use CACTI-3DD [34] and Micron's memory system power calculator [48, 49]. Table 2 shows die area and row activation energy breakdown of 2Gb x8 DDR3-1600 chip, which is obtained by CACTI-3DD simulation at the 20nm technology node (same technology node as [11]). Through this result, it is obvious that the row activation power and energy are mainly consumed on the local bitlines and the local sense amplifiers as discussed in Section 2.2. In other words, activation power is proportional to the number of bitlines in MATs being activated during a memory access. Figure 9 shows the energy proportionality to the number of MATs activated. As we can see in Figure 9, the activation energy is dramatically reduced with less MATs. However, the energy reduction cannot reach 50% even though reducing MATs by half because of shared structures (i.e., row activation bus and row predecoder) across MATs in a sub-array. We take this into account for the power model of partial row activation and project scaling factors of activation energy onto the industrial power consumption parameter (i.e., P_{ACT}). To estimate pure activation power consumption based on the industrial reports [11, 48], we use Equation 1 and Equation 2 below, where *IDD0* is the row activation current during a row cycle ${}^tRC$, and *IDD2N* and *IDD3N* are the background currents for the situation when all banks are idle or at least one bank is active, respectively. Note that row cycle (${}^tRC$) is the sum of row activation time (${}^tRAS$) and bank precharge time (${}^tRP$). Thus, actual row activation power consumption includes power consumptions of pure row activation and background (active and idle during row cycle) as well as bank precharge. The estimated row activation power parameters at full, 7/8th, 6/8th rows, and so forth are given in Table 3.

$$I_{ACT} = IDD0 - \frac{IDD3N \times {}^tRAS + IDD2N \times ({}^tRC - {}^tRAS)}{{}^tRC} \quad (1)$$

$$P_{ACT} = VDD \times I_{ACT} \quad (2)$$

5.1.2 Experimental Setup

We evaluate our proposed scheme using a cycle-accurate integrated CPU-DRAM simulation platform, gem5 [50] and DRAMSim2 [51]. The detailed baseline system configuration is described in Table 3. We use the FR-FCFS request scheduler [16] with row-interleaved address mapping and separate 64-entry read/write request queues. In addition, we use the relaxed close-page policy with precharge power-down, which closes the opened row when there is no pending request in the request queues that can benefit from row buffer hit and saves background power consumption during idle. To prevent request starvation and fairness problem among requests, we restrict accesses to an opened row up to four requests [15]. We also include the results of restricted close-page policy, in which every read and write request involves a pair of row activation and bank precharge. We use line-interleaved address mapping for the restricted close-page policy, which maximizes available parallelisms of the DRAM system.

For workloads, we use mixes of several applications from SPEC CPU2006 (*bzip2*, *lbm*, *libquantum*, *mcf*, *omnetpp*) [52] and Olden (*em3d*) [53] benchmark suites as well as the *GUPS* [54] and *LinkedList* [55] microbenchmarks. For cycle-based simulations, we run 200 million instructions from a representative region from each application. We use SimPoint [56] for SPEC CPU2006 applications to determine the representative regions and manually skip the initialization phase for the regularly-behaved Olden benchmark suite and microbenchmarks. Our collection of benchmarks is primarily memory intensive but also includes a compute-bound application (*bzip2*). We use four identical instances of single-threaded applications and also run the application mixes described in Table 4.

Table 3: Baseline system configuration.

Processor	
Cores (4-core)	3.2GHz x86 8-way superscalar out-of-order, LDQ/STQ/ROB 32/32/192
L1 I/D caches	32kB, 4-way associative, 2-cycle latency, 64B cache line
L2 cache (shared)	4MB, 8-way associative, 20-cycle latency, 64B cache line
DRAM system	
Organization	8GB, 2 channels, 2 ranks per channel, 8 chips per rank, 64-bit data bus
Memory controller	FR-FCFS scheduler, row-interleaved mapping, relaxed close-page policy w/ precharge power-down 64/64-entry read/write request queues per channel (48/16 high/low watermarks for write request queue)
DRAM chip	
Organization	2Gb DDR3-1600 x8, bank/row/column 8/32k/1k
Timing (cycle)	^{t}RCD (11/**12**), ^{t}RP (11), ^{t}CAS (11), ^{t}RAS (28), ^{t}WR (12), ^{t}CCD (4), ^{t}RRD (5), ^{t}FAW (24), ^{t}RC (39)
Power (mW)	PRE STBY (27), PRE PDN (18), REF (210), ACT STBY (42), RD (78), WR (93), RD I/O (4.6), WR ODT (21.2), RD/WR TERM (15.5/15.4), ACT full, 7/8, 6/8, 5/8, 4/8, 3/8, 2/8, and 1/8 row (**22.2**, **19.6**, **16.9**, **14.3**, **11.6**, **9.1**, **6.4**, **3.7**)

Table 4: Workload mixes.

MIX1	bzip2, lbm, libquantum, omnetpp
MIX2	mcf, em3d, GUPS, LinkedList
MIX3	bzip2, mcf, lbm, em3d
MIX4	libquantum, GUPS, omnetpp, LinkedList
MIX5	bzip2, LinkedList, lbm, GUPS
MIX6	libquantum, em3d, omnetpp, mcf

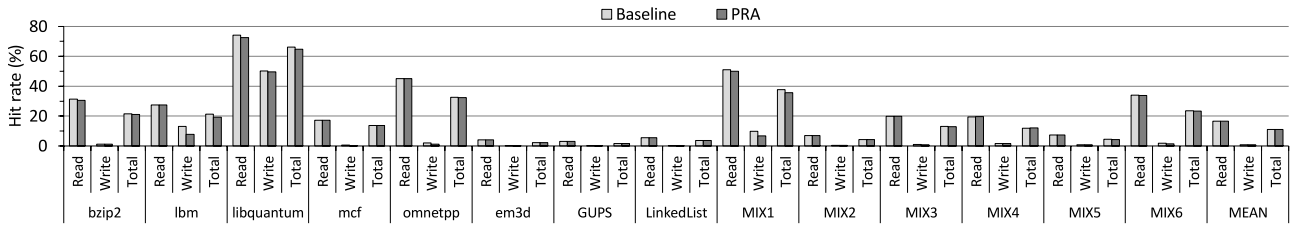
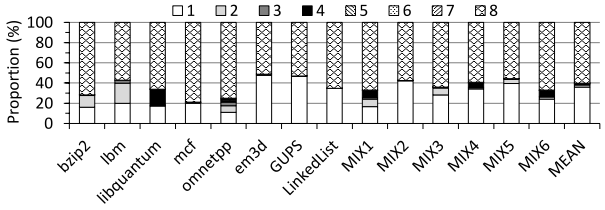
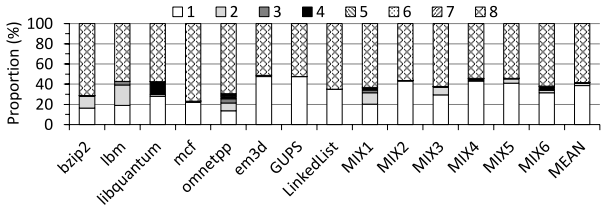


Figure 10: PRA imact on row buffer read, write, and total hit rates.



(a) Restricted close-page policy.



(b) Relaxed close-page policy.

Figure 11: Proportion of row activation granularities with PRA.

As the performance metric, we use *weighted speedup* (WS) defined in Equation 3, where N is the number of cores in the CMP, IPC_i^{shared} is the IPC of the i-th application when running with other applications, and IPC_i^{alone} is the IPC of the i-th application when running alone in the CMP. We denote normalized weighted speedup as *normalized performance* in the remaining experimental results.

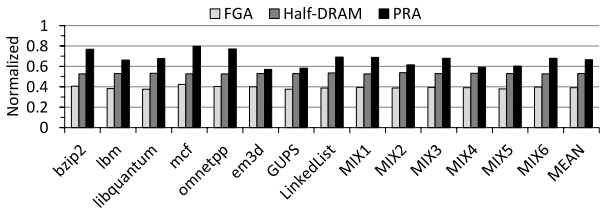
$$WS = \sum_{i=1}^{N} \frac{IPC_i^{shared}}{IPC_i^{alone}} \quad (3)$$

5.2 Results and Discussion

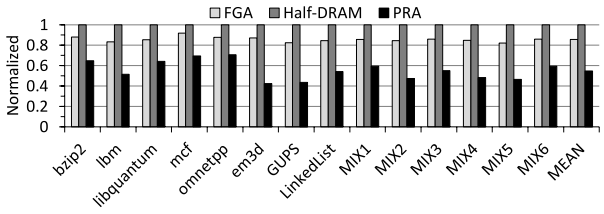
5.2.1 Impact of Partial Row Activation

With the relaxed close-page policy that closes an opened row when there is no pending request in the request queues, PRA can cause a row buffer miss when read and write accesses target the opened row but the requested memory block is not activated because the row is partially opened. We define this event as *false row buffer hit*, which is supposed to be a *hit* in the conventional DRAM architecture but turned out a *miss* in the PRA architecture. On a false row buffer hit, the opened row should be closed first and then row activation has to be conducted for an incoming access. For example, if the PRA mask is 11000000b and thus local rows of the first and second groups of MATs are currently open, a posterior read request that targets the partially opened row will result in a false row buffer hit because a full row should be activated for the read request. This involves a pair of activation and precharge operations which are not supposed to occur in the conventional DRAM architecture. As a result, an upcoming request that will experience a false row buffer hit can be delayed by a series of precharge and activation operations. Figure 10 shows the impact of PRA on row buffer hit rates classified into read, write, and combined cases. We count false row buffer hits as row buffer misses and a pair of precharge and activation is performed prior to a request that results in a false row buffer hit in our experiments. As shown in Figure 10, false row buffer hits on read requests are rare, which indicates that most read requests are not delayed owing to parially opened rows. Only up to 0.26% and average 0.04% of read requests experience false hits in our experiments. As we discussed in Section 2.2, this is because there is significant locality asymmetry between read and write traffics. A false row buffer hit can also occur in a write case. If currently opened row is maintained by 10000001b PRA mask, an incoming write request that needs a local row of the second group of MATs will result in a false hit. In case of that multiple requests heading to the same row with the different PRA masks exist in the request queue, PRA masks are ORed to activate partial rows as many as possible to accommodate all requests targeting the same row. As we can see in Figure 10, *lbm* and *libquantum* benchmarks show high row buffer write hit rates. In *libquantum*, almost all write requests hit partially opened rows, which does not lower the row buffer write hit rate. On the other hand, the row buffer write hit rate is lowered by 5% (from 13% to 8%) due to 50% false hit rate in the *lbm* benchmark. Nevertheless, its impact on performance is insignificant because write requests are not urgent requests and thus generally scheduled in background to help the read bandwidth. However, the power benefits of PRA can be slightly reduced owing to extra row activations caused by false hits. Specifically, total row buffer hit rate is reduced by 0.1% on average (from 11.2% to 11.1%) by false row buffer hits.

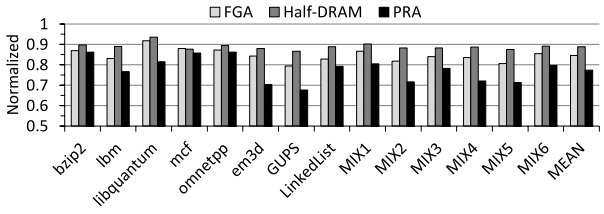
Figure 11 shows the proportion of row activation granularities when relaxed and restricted close-page policies are used. With the restricted close-page policy shown in Figure 11(a), the word-level dirtiness of a cache line as shown in Figure 3 directly reflects row activation granularity because the DRAM access is composed of a row activation, column access, and bank precharge atomically in the restricted close-page policy. With the relaxed close-page policy shown in Figure 11(b), the proportion of a full row activation is reduced in applications that show high row buffer read hit rate. In a such case, less row activations occur by read requests in which row activations caused by write requests have more



(a) Row activation power consumption.

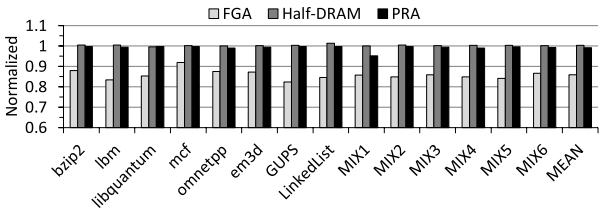


(b) I/O power consumption.

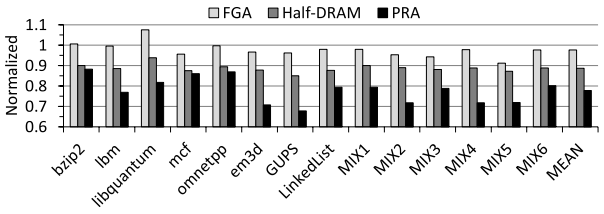


(c) Total power consumption.

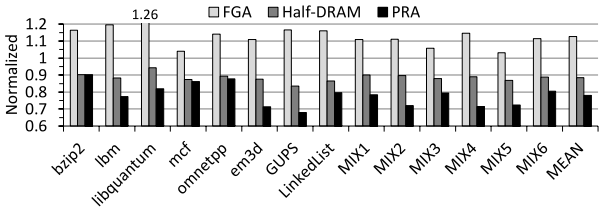
Figure 12: Normalized DRAM row activation, I/O (including read I/O, write ODT, and read/write termination), and total power consumptions of FGA, Half-DRAM, and PRA.



(a) Normalized performance.



(b) DRAM energy consumption.



(c) Energy-delay product (EDP).

Figure 13: Normalized performance, DRAM energy consumption, and energy-delay product of FGA, Half-DRAM, and PRA.

impact on the power consumption of row activation as shown in Table 1. Specifically, the average proportions of row activation granularities from a one-eighth to a full row are 39%, 2%, 0.43%, 0.45%, 0.05%, 0.05%, 0.02%, and 58%, respectively, with the relaxed close-page policy. Those numbers with the restricted close-page policy are 36%, 2.3%, 0.4% 1.2%, 0.04%, 0.04%, 0.02%, and 60%, respectively. As we can see from these results, a large portion of write requests necessitates only small portion of cache line written to DRAM and partial row activation for these small granularity writes significantly reduces row activation granularity.

5.2.2 Performance and Power Consumption

Figure 12 shows the DRAM row activation, I/O, and total power consumptions, and its breakdown for FGA, Half-DRAM, and PRA. The results are normalized to those of the baseline in which the relaxed close-page policy is used. Because FGA at the one-eighth row granularity incurs severe performance degradation, FGA at the half row granularity is chosen for experiments. For Half-DRAM, we use Half-DRAM-1Row scheme in [10] to see the pure impact of the Half-DRAM scheme because the other scheme of Half-DRAM, Half-DRAM-2Row is the integrated scheme with the sub-array level parallelism [14] scheme. As shown in Figure 12(a), PRA achieves significant reduction in row activation power consumption because PRA can benefit from even one-eighth row activation. The row activation power consumption is reduced by up to 43% and 34% on average. FGA and Half-DRAM achieve more row activation power saving than PRA because both read and write traffics are covered and a half row is always activated for all the traffics. In addition to row activation power saving, PRA consumes up to 58% and average 45% less I/O power because only necessary data heading to partial rows are transferred to DRAM. As shown in Figure 12(b), a large amount of I/O power consumption can be reduced in PRA because write ODT power consumption is much larger than read I/O and read/write termination power consumptions [7, 26, 48, 49]. In contrast, Half-DRAM and FGA should transfer the entire cache line to/from DRAM as conventional DRAM system does, which shows the same I/O power dissipation as baseline. Note that I/O power reduction in FGA is due to increased runtime (16 bursts in 8 memory cycles are taken to transfer a 64B cache line to/from a rank, which is supposed to be 8 bursts in 4 memory cycles in the conventional DRAM architecture). As shown in Figure 12(c), because PRA can effectively reduce both row activation and I/O power consumptions, total DRAM power consumption is reduced by up to 32% and 23% on average, which outperforms FGA and Half-DRAM in DRAM power saving. In FGA and Half-DRAM, total DRAM power consumption is reduced by 15% and 11% on average, respectively.

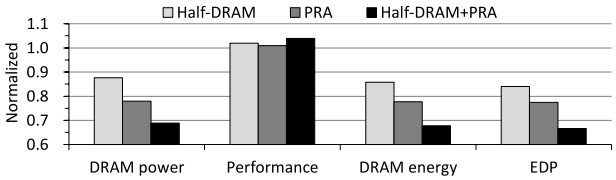


Figure 14: Average DRAM power, normalized performance, DRAM energy, and EDP of Half-DRAM, PRA, and combined scheme (Half-DRAM + PRA).

Figure 13 shows the normalized performance, DRAM energy consumption, and energy-delay product (EDP) of FGA, Half-DRAM, and PRA. As shown in Figure 13(a), the relaxation of ^tRRD and ^tFAW timing constraints is countervailed by the timing overheads of PRA (i.e., ^tRCD+^tCK and false row buffer hits). The timing overheads of PRA have little impact on the performance because extra one memory cycle for PRA mask delivery is only required for write row activations and false row buffer hits rarely occur in most applications. Specifically, PRA shows the performance loss by up to 4.8% and 0.8% on average. Because ^tRRD and ^tFAW timing constraints can be relaxed in both read and write row activations, Half-DRAM shows up to 1.5% and average 0.3% performance improvement. On the other hand, FGA incurs significant performance loss even with the half row activation. The performance loss of FGA is estimated to be up to 18% and 14% on average. As shown in Figure 13(b) and 13(c), PRA shows the best savings in DRAM energy consumption and EDP compared to FGA and Half-DRAM. PRA is shown to reduce DRAM energy consumption by up to 34% and 23% on average, which results in maximum 32% and average 22% EDP reduction.

5.2.3 A Case Study of PRA with Other Schemes

We evaluate the impact of PRA with the state-of-the-art schemes, Half-DRAM [10] and the Dirty-block index (DBI) [31] addressing the overfetching problem of intra-chip row activation and improving the overfetching efficiency of write row activation, respectively.

PRA with Half-DRAM. Half-DRAM is designed to vertically divide MATs into two halves rather than selecting or deselecting MATs for row activation. Because of its different organizations from PRA, Half-DRAM can be combined with PRA together for further DRAM power saving and performance improvement. This can be implemented by embedding PRA latches and wordline gates on top of Half-DRAM organization to enable selecting or deselecting MATs in odd and even groups for write row activations. Figure 14 shows the impact of Half-DRAM, PRA, and the combined scheme (Half-DRAM + PRA), which is averaged from the results of all 14 benchmarks. To see the best impact on the performance and power consumptions, we use the restricted close-page policy for this study. As shown in Figure 14, the combined scheme shows synergy in all results. Both PRA and Half-DRAM benefit from relaxed timing constraints that have more impact on the performance in the restricted close-page policy. Thus, the combined scheme shows better performance improvement compared to individual schemes. Moreover, significant reduction is achieved

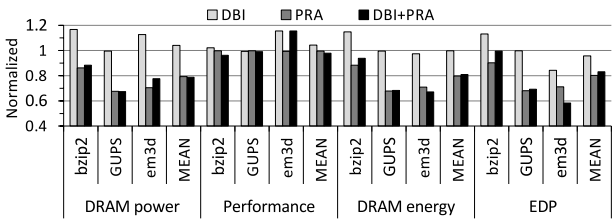


Figure 15: DRAM power, normalized performance, DRAM energy, and EDP of DBI, PRA, and combined scheme (DBI + PRA). MEAN is average of all 14 benchmarks.

in DRAM power/energy consumption and EDP because the combined scheme benefits from half read/write row activations by Half-DRAM and finer-granularity write row activations with reduced I/O power consumption by PRA.

PRA with DBI. DBI proactively writes back dirty cache lines of a DRAM row when any dirty line from that row is evicted from the cache. Thus, the row buffer hit rate of write traffic can be improved by DBI. Figure 15 shows the results of DBI, PRA, and the combined scheme (DBI + PRA). Because the combined scheme shows dynamic behavior in performance and DRAM power consumption, we choose several benchmarks which show the representative results; *bzip2* is the case where DRAM power saving is obtained by PRA but the performance gain of DBI is lost, *GUPS* only benefits from PRA, and *em3d* is the case where both PRA and DBI produce synergy. On average, the combined scheme shows better results than DBI alone but worse result than PRA only scheme owing to increased false row buffer hits. Because DBI produces intensive write traffic, the probability of false row buffer hits increases although PRA masks are likely to be merged if successive writes in the request queue are heading to the same row. As we can see through this case study, DBI has more impact on the performance improvement whereas PRA is more effective in DRAM power saving.

6. CONCLUSION

The modern DRAM architecture is designed to provide wide bandwidth and low latency by placing a large number of cells in rows of independently accessible banks. Owing to this characteristic, the row overfetching problem arises in which a several-KB row has to be activated even for a small data request (e.g., a 64B cache line). In addition, I/O energy is a major component of the overall DRAM energy consumption and expected to become an even more critical problem in future systems due to the increasing demand of off-chip memory bandwidth. In this paper, we present PRA, a partial row activation technique addressing the row overfetching problem of modern DRAM architecture and to reduce the power consumption of DRAM row activation and write I/O switching. PRA exploits the inherent fine-grained structures of DRAM (i.e., MATs) and makes them activated or deactivated on a row activation request. Unlike prior studies [8, 9, 10], PRA enables various granularity row activation ranging from a full row to one-eighth row without sacrificing the bandwidth of DRAM through the asymmetric row activation mechanism for read and write requests. According to our experiments, PRA is shown to reduce the row activation and

I/O power consumptions by 34% and 45% on average, respectively, which results in average 23% total DRAM power saving with little performance loss and small hardware overheads. Therefore, a partial row activation can be a promising and cost-effective approach for low-power DRAM systems.

ACKNOWLEDGEMENT

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean Government (No. 2014R1A2A2A01007051) and by IDEC (EDA Tool).

REFERENCES

[1] U. Hoelzle and L. A. Barroso, *The Datacenter As a Computer: An Introduction to the Design of Warehouse-Scale Machines*. Morgan and Claypool Publishers, 2009.

[2] K. Lim, J. Chang, T. Mudge, P. Ranganathan, S. K. Reinhardt, and T. F. Wenisch, "Disaggregated Memory for Expansion and Sharing in Blade Servers," in *ISCA*, 2009.

[3] K. T. Malladi, B. C. Lee, F. A. Nothaft, C. Kozyrakis, K. Periyathambi, and M. Horowitz, "Towards Energy-Proportional Datacenter Memory with Mobile DRAM," in *ISCA*, 2012.

[4] D. H. Yoon, J. Chang, N. Muralimanohar, and P. Ranganathan, "BOOM: Enabling Mobile Memory Based Low-Power Server DIMMs," in *ISCA*, 2012.

[5] H. Zheng, J. Lin, Z. Zhang, and Z. Zhu, "Decoupled DIMM: Building High-bandwidth Memory System Using Low-speed DRAM Devices," in *ISCA*, 2009.

[6] H. David, C. Fallin, E. Gorbatov, U. R. Hanebutte, and O. Mutlu, "Memory Power Management via Dynamic Voltage/Frequency Scaling," in *ICAC*, 2011.

[7] Y. Song and E. Ipek, "More is Less: Improving the Energy Efficiency of Data Movement via Opportunistic Use of Sparse Codes," in *MICRO*, 2015.

[8] A. N. Udipi, N. Muralimanohar, N. Chatterjee, R. Balasubramonian, A. Davis, and N. P. Jouppi, "Rethinking DRAM Design and Organization for Energy-Constrained Multi-Cores," in *ISCA*, 2010.

[9] E. Cooper-Balis and B. Jacob, "Fine-Grained Activation for Power Reduction in DRAM," *IEEE Micro*, 2010.

[10] T. Zhang, K. Chen, C. Xu, G. Sun, T. Wang, and Y. Xie, "Half-DRAM: A High-bandwidth and Low-power DRAM Architecture from the Rethinking of Fine-grained Activation," in *ISCA*, 2014.

[11] Samsung Electronics Co., Ltd., "2Gb E-die DDR3 SDRAM Datasheet – K4B2G0846E." `http://www.samsung.com/semiconductor/products/dram/server-dram`.

[12] B. Keeth, R. J. Baker, B. Johnson, and F. Lin, *DRAM Circuit Design: Fundamental and High-Speed Topics*. Wiley-IEEE Press, 2007.

[13] T. Vogelsang, "Understanding the Energy Consumption of Dynamic Random Access Memories," in *MICRO*, 2010.

[14] Y. Kim, V. Seshadri, D. Lee, J. Liu, and O. Mutlu, "A Case for Exploiting Subarray-Level Parallelism (SALP) in DRAM," in *ISCA*, 2012.

[15] D. Kaseridis, J. Stuecheli, and L. K. John, "Minimalist Open-page: A DRAM Page-mode Scheduling Policy for the Many-core Era," in *MICRO*, 2011.

[16] S. Rixner, W. J. Dally, U. J. Kapasi, P. Mattson, and J. D. Owens, "Memory Access Scheduling," in *ISCA*, 2000.

[17] Y. Lee, S. Kim, S. Hong, and J. Lee, "Skinflint DRAM System: Minimizing DRAM Chip Writes for Low Power," in *HPCA*, 2013.

[18] D. H. Yoon, M. K. Jeong, M. Sullivan, and M. Erez, "The Dynamic Granularity Memory System," in *ISCA*, 2012.

[19] F. Ware and C. Hampel, "Improving Power and Data Efficiency with Threaded Memory Modules," in *ICCD*, 2006.

[20] H. Zheng, J. Lin, Z. Zhang, E. Gorbatov, H. David, and Z. Zhu, "Mini-Rank: Adaptive DRAM Architecture for Improving Memory Power Efficiency," in *MICRO*, 2008.

[21] J. H. Ahn, N. P. Jouppi, C. Kozyrakis, J. Leverich, and R. S. Schreiber, "Future Scaling of Processor-memory Interfaces," in *SC*, 2009.

[22] J. H. Ahn, J. Leverich, R. Schreiber, and N. P. Jouppi, "Multicore DIMM: An Energy Efficient Memory Module with Independently Controlled DRAMs," *CAL*, 2009.

[23] D. H. Yoon, M. K. Jeong, and M. Erez, "Adaptive Granularity Memory Systems: A Tradeoff Between Storage Efficiency and Throughput," in *ISCA*, 2011.

[24] N. D. Gulur, R. Manikantan, M. Mehendale, and R. Govindarajan, "Multiple Sub-row Buffers in DRAM: Unlocking Performance and Energy Improvement Opportunities," in *ICS*, 2012.

[25] Y. H. Son, O. Seongil, H. Yang, D. Jung, J. H. Ahn, J. Kim, J. Kim, and J. W. Lee, "Microbank: Architecting Through-Silicon Interposer-Based Main Memory Systems," in *SC*, 2014.

[26] H. Seol, W. Shin, J. Jang, J. Choi, J. Suh, and L.-S. Kim, "Energy Efficient Data Encoding in DRAM Channels Exploiting Data Value Similarity," in *ISCA*, 2016.

[27] K. Basu, A. Choudhary, J. Pisharath, and M. Kandemir, "Power Protocol: Reducing Power Dissipation on Off-Chip Data Buses," in *MICRO*, 2002.

[28] M. R. Stan and W. P. Burleson, "Bus-Invert Coding for Low-Power I/O," *TVLSI*, 1995.

[29] C. J. Lee, V. Narasiman, E. Ebrahimi, O. Mutlu, and Y. N. Patt, "DRAM-Aware Last-Level Cache Writeback: Reducing Write-Caused Interference in Memory Systems," Tech. Rep. TR-HPS-2010-2, University of Texas at Austin, 2010.

[30] J. Stuecheli, D. Kaseridis, D. Daly, H. C. Hunter, and L. K. John, "The Virtual Write Queue: Coordinating DRAM and Last-Level Cache Policies," in *ISCA*, 2010.

[31] V. Seshadri, A. Bhowmick, O. Mutlu, P. B. Gibbons, M. A. Kozuch, and T. C. Mowry, "The Dirty-block Index," in *ISCA*, 2014.

[32] J. S. Liptay, "Structural Aspects of the System/360 Model 85: II the Cache," *IBM Systems Journal*, 1968.

[33] Samsung Electronics Co., Ltd., "8Gb B-die DDR4 SDRAM Datasheet – K4A8G085WB." `http://www.samsung.com/semiconductor/products/dram/server-dram`.

[34] K. Chen, S. Li, N. Muralimanohar, J. H. Ahn, J. B. Brockman, and N. P. Jouppi, "CACTI-3DD: Architecture-level Modeling for 3D Die-stacked DRAM Main Memory," in *DATE*, 2012.

[35] S. W. Keckler, W. J. Dally, B. Khailany, M. Garland, and D. Glasco, "GPUs and the Future of Parallel Computing," *IEEE Micro*, 2011.

[36] B.-S. Kong, S.-S. Kim, and Y.-H. Jun, "Conditional-Capture Flip-Flop for Statistical Power Reduction," *JSSC*, 2001.

[37] K. Itoh, *VLSI Memory Chip Design*. Springer, 2001.

[38] B. Schroeder, E. Pinheiro, and W.-D. Weber, "DRAM Errors in the Wild: A Large-Scale Field Study," in *SIGMETRICS*, 2009.

[39] S. Lin and D. Costello, *Error Control Coding: Fundamentals and Applications*. Pearson-Prentice Hall, 2004.

[40] T. J. Dell, "A White Paper on the Benefits of Chipkill-Correct ECC for PC Server Main Memory," *IBM Microelectronics Division*, 1997.

[41] Samsung Electronics Co., Ltd., "240pin DDR3 Unbuffered DIMM Datasheet – M391B5773DH0." `http://www.samsung.com/semiconductor/products/dram/pc-dram`.

[42] Samsung Electronics Co., Ltd., "240pin DDR3 Registered DIMM Datasheet – M393B5773DH0." `http://www.samsung.com/semiconductor/products/dram/server-dram`.

[43] A. Seznec, "Decoupled Sectored Caches: Conciliating Low Tag Implementation Cost," in *ISCA*, 1994.

[44] A. González, C. Aliagas, and M. Valero, "A Data Cache with Multiple Caching Strategies Tuned to Different Types of Locality," in *ICS*, July 1995.

[45] J. B. Rothman and A. J. Smith, "The Pool of Subsectors Cache Design," in *ICS*, 1999.

[46] S. Kumar and C. Wilkerson, "Exploiting Spatial Locality in Data Caches Using Spatial Footprints," in *ISCA*, 1998.

[47] C. F. Chen, S. H. Yang, B. Falsafi, and A. Moshovos, "Accurate and Complexity-Effective Spatial Pattern Prediction," in *HPCA*, 2004.

[48] Micron Inc., "TN-41-01: Calculating Memory System Power for DDR3." `http://www.micron.com/support/power-calc`.

[49] Micron Inc., "DDR3 SDRAM System-Power Calculator." `http://www.micron.com/support/power-calc`.

[50] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The gem5 simulator," *CAN*, 2011.

[51] P. Rosenfeld, E. Cooper-Balis, and B. Jacob, "DRAMSim2: A Cycle Accurate Memory System Simulator," *CAL*, 2011.

[52] Standard Performance Evaluation Corp., "SPEC CPU2006 Benchmark Suite." `https://www.spec.org/cpu2006`.

[53] M. C. Carlisle and A. Rogers, "Software Caching and Computation Migration in Olden," in *PPoPP*, 1995.

[54] B. R. Gaeke, "GUPS (Giga-Updates per Second) Benchmark." `http://www.dgate.org/~brg/files/dis/gups`.

[55] C. Zilles, "Linked List Traversal Micro-Benchmark." `http://zilles.cs.illinois.edu/llubenchmark.html`.

[56] G. Hamerly, E. Perelman, J. Lau, and B. Calder, "SimPoint 3.0: Faster and More Flexible Program Analysis," in *MoBS*, 2005.