Lazy Precharge: An Overhead-free Method to Reduce Precharge Overhead for Memory Parallelism Improvement of DRAM System

Tao Zhang, Cong Xu and Yuan Xie
Department of Computer Science and Engineering
The Pennsylvania State University, PA, USA 16802
Email: {zhangtao, czx102, yuanxie}@cse.psu.edu

Guangyu Sun
School of Electrical Engineering and Computer Science
Peking University, Beijing, China 100871
Email: gsun@pku.edu.cn

***Abstract*—As we enter the multi-core era, the main memory becomes the bottleneck due to the exploded memory requests. In this work, we propose a novel memory architecture–*Lazy Precharge* (LaPRE) that enables aggressive activation schemes so that multiple rows in a bank can be activated successively without the interrupt from precharges. Therefore, LaPRE effectively reduces the precharge overhead and thus improves memory parallelism. In addition, three memory scheduling schemes are proposed correspondingly to fully make use of the improved memory parallelism. The experimental results show that LaPRE can achieve 14% performance improvement on average without hardware overhead.**

I. Introduction

As more and more transistors can be integrated into a single chip, the rich transistor resource enables chip multi-processor (CMP) architecture, which in turn produces heavy memory loads and therefore generates tremendous stress on the main memory (a.k.a."memory wall"). On the other hand, the memory requests from multiple cores render high random behavior so that low row buffer locality can be exploited [1], [2]. The poor spatial locality makes it more challenging to design a memory controller with satisfied performance. Given the low row buffer locality, the *Close-Page* policy that closes the entire row buffer immediately after the data access is preferable in the CMP.

The memory access latency in a memory system with Close-Page policy is mainly determined by the *Row Cycle* (tRC). Unfortunately, the row cycle is almost constant through the JEDEC-DDR evolutions (DDR-DDR4). Considering the memory clock frequency doubles in each generation, the unchanged row cycle indicates the memory becomes relatively slower (in cycle). For example, the row cycle of DDR3-1600 is 41 cycles while it becomes 54 cycles in DDR4-2400 [3], [4]. Intuitively, the larger memory latency suppresses the memory level parallelism since a bank must spend more cycles to deliver the data. To compensate the reduced memory parallelism, sub-array level parallelism (SALP) [5] has been proposed to hide the latency with fine-grained memory structure and tiered-latency DRAM (TL-DRAM) [6] have been studied to shorten the row cycle in the *Near segment* of a bank. Both prior arts, however, requires additional logic in the memory chip, which leads to increasing cost per bit ($/bit). Distinguished from those work, we propose a simple but effective scheme, named *Lazy Precharge* (LaPRE), to smartly leverage the legacy of existing sub-array infrastructure to reduce the precharge overhead as well as the row cycle. In summary, our contributions in this work are:

• We propose LaPRE as an effective solution to reduce the precharge overhead and thus shorten the row cycle for performance improvement. LaPRE does not incurs any design overhead in the memory chip. The zero overhead makes LaPRE promising to be adopted by the industry.

• We propose three memory scheduling schemes in the memory controller (MC) to leverage LaPRE to obtain performance improvement. The impact of address mapping schemes and power constraint are also evaluated as sensitivity study.

This work is supported in part by SRC grants, NSF 0903432 and 1017277, NSF China 61202072, National High-tech R&D Program of China 2013AA013201, and AMD Gift Grant.

II. Background

A. DRAM Hierarchy

To better understand the enabling techniques of LaPRE, we first review the conventional DRAM architecture that is widely used in commodity memory. Without loss of generality, the DRAM memory structure has a pyramid-like hierarchy, which, as shown in Figure 1 (from top to bottom), consists of rank, chip, bank, sub-array, MAT, and cell. A rank is composed of multiple memory chips (a.k.a. device) that operate in lockstep to feed the data bus. Inside one chip, several banks are employed as cell arrays and can be accessed independently. Therefore, bank-level parallelism is extensively studied to improve memory performance.

In fact, a bank can be further divided into many sub-arrays. All sub-arrays share the output of global row address decoder so that only one sub-array is allowed to be active at any time. In one sub-array, there are many MATs and each of them has its own row decoder and sense amplifier array as the local row buffer. Typically, a MAT is sized of 512×512 storage cells in row (wordline) and column (bitline) dimension. One storage cell is a **C**apacitor that is connected to an access **T**ransistor (so-called 1T1C structure). According to the different voltage in the capacitor, the cell holds logic '1' (V_{dd}) or '0' (0V). As the example shown in the figure, eight chips compose a rank to provide 64-bit data and each chip delivers 8-bit data (so-called ×8 chip). Every chip contains eight banks and each bank has 16K rows and 8K columns. Therefore, one bank has 32 ($=\frac{16,384}{512}$) sub-arrays and each sub-array consists of 16 ($=\frac{8,192}{512}$) MATs. In this work, LaPRE falls into sub-array level parallelism (SALP), which allows more than one sub-arrays to be active and those sub-arrays share a common precharge.

B. Basic Operations in DRAM

The basic operations of a sub-array can be classified into four types[1]: row activation (`ACT` or `RAS`), column read/write (`CAS-R/W`),

[1]In fact, more memory commands are available in JEDEC specification, such as implicit auto precharge, post-CAS, and commands for low-power mode. The PRE-ACT-CAS-PRE command order, however, is still applied. As a result, we only show the basic flow in this paper.

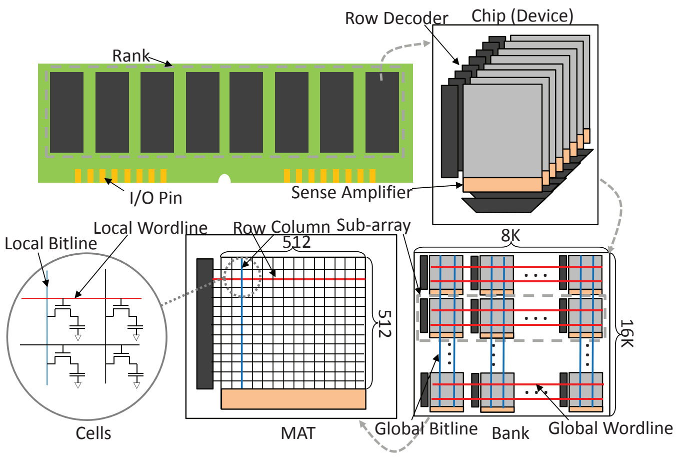


Fig. 1: DRAM hierarchy – a 1Gb-8bank×8 example

precharge (PRE), and refresh (REF). Only CAS has data transfer between MC and memory modules while other three operations are only used for either sub-array access preparation (ACT and PRE) or data protection (REF). In this work, we focus on ACT and PRE operations. As DRAM depends on electrical (dis-)charging, we explain each operation with its unique electrical behavior as follows.

As shown in Figure 2(1), at the beginning all sub-arrays in a bank are precharged as the signal *EQ* in the precharge logic (equilibration logic) is asserted. Therefore, the voltage of bitline (BL) is driven to V$_{dd}$/2 during idle. Once an ACT command is coming, *EQ* becomes inactive to prevent the precharge logic from driving the bitline (so not shown in Figure 2②③). The sub-array then goes through *wordline opening*, *charge sharing*, *data sensing*, and *data restoring* in sequence. First of all, the row address decoder asserts one wordline based on the row address as shown in Figure 2(2) (red line). Correspondingly, all access transistors in the row are open. Afterwards, the sub-array enters charge sharing. If the voltage of storage cell is V$_{dd}$ (0V), the cell (bitline) shares charges with the bitline (cell). After the charge sharing, the bitline has a positive (negative) voltage difference δV from the reference bitline that is still under V$_{dd}$/2. Next, in the sensing logic signal *ACT* goes to V$_{dd}$ and *NLAT* goes to 0V for the data sensing. The sensing logic in the sense amplifier (SA) can recognize the voltage difference and amplifies the sensing result to '1' ('0')(Figure 2(2)(3)). After the data sensing, the data is locked in the SA for the following data burst by CAS-R/W. Note that the original data in the cell has been destroyed due to the charge sharing. As a result, SA needs to drive the bitline to restore the data into the cell[2].

It is obvious that the bitline voltage is no longer V$_{dd}$/2 after an activation. As a consequence, no further ACTs are permitted in the active sub-array since SA may not work correctly. To open another row in the same sub-array, a PRE must be issued at first shown in Figure 2(4). A PRE mainly does two things: wordline closing and bitline/SA reset. Once there is a PRE command, the row address decoder desserts the wordline so that the access transistors in the row are completely closed. Then, signal *EQ* becomes active again so that the precharge logic drives the bitline back to V$_{dd}$/2 for the next ACT. In the commodity DRAM, all sub-arrays in a bank share signal *EQ* so that one PRE precharges the whole bank even though in fact only one sub-array is precharged. Note that a precharge can only be issued after the completion of data restoring since the wordline closing can early terminate the restore and thus results in data loss. As no data transfer occurs during a precharge, it inevitably induces performance overhead in DRAM.

III. Design of Lazy Precharge

In this section, we first define the overhead of percharge with Close-Page policy as the motivation. The design of LaPRE is then presented and the corresponding memory scheduling policies are extended to Open-Page policy.

A. Overhead of Precharge

Figure 3 (top) shows the timeline of three requests to the same bank in the baseline. Even though they refer to different sub-arrays, all requests have to be serviced in sequence as we assume no SALP is applied. In particular, every request has a dedicated precharge due to the Close-Page policy. According to DRAM timing constraint, we define *overhead of precharge* (OH_{PRE}) as the ratio of precharge time in a row cycle as shown in Eq.1 and 2, where the overhead for read and write is calculated separately. Based on the values of timing parameters in Table I, the overhead of precharge can be up to 30% and 25.6% for read and write, respectively. The large overhead indicates an opportunity to improve the memory performance by reducing the precharge overhead.

$$OH_{PRE}^{read} = \frac{tRP}{tRAS + tRP} = \frac{tRP}{tRC} \tag{1}$$

$$OH_{PRE}^{write} = \frac{tRP}{tRCD + tCWD + tBURST + tWR + tRP} \tag{2}$$

In fact, SALP-1 has been proposed in [5] to reduce the precharge overhead. As shown in the middle of Figure 3, SALP-1 relaxes the precharge overhead by overlapping the precharge of one sub-array with the activation of another sub-array. SALP-1, however, incurs the conflict between activation and precharge in a bank for two reasons. The first reason is the conflict of wordline opening and closing in the sub-array that an activation is undergoing. As mentioned in [5] (Section 5.2), "... a PRECHARGE is designed to lower all wordlines within a bank to zero voltage..." while an activation should assert one of the wordlines. The second reason is that all sub-arrays are precharged simultaneously resulting from the sharing of signal *EQ*. As a consequence, SALP-1 is in fact impractical to be implemented without extra logics for selective sub-array-level precharge, like SALP-2 or MASA in that paper.

TABLE I: Timing Parameters from Micron Data sheet [3]

Timing Params.	Value	Timing Params.	Value
tRAS	36ns	tRP	15ns
tRCD	15ns	tCWD	7.5ns
tBURST	6ns	tWR	15ns

Distinct from SALP-1, LaPRE eliminates both conflicts by removing the overlap of activation and precharge. Instead, LaPRE delays the precharge and allows multiple activations before a coalesced precharge, which is named *Lazy Precharge* (LaPRE). Figure 3 (bottom) illustrates the basic idea behind LaPRE. As shown, the ACT to sub-array 2 (3) is able to start immediately after the data has been restored in the prior active sub-array 1 (2). At the end, a lazy precharge is issued to precharge all three sub-arrays. In this way, only one precharge is needed compared to the baseline that requires three precharges.

[2]The reader can refer to [7] for more details of DRAM operations.

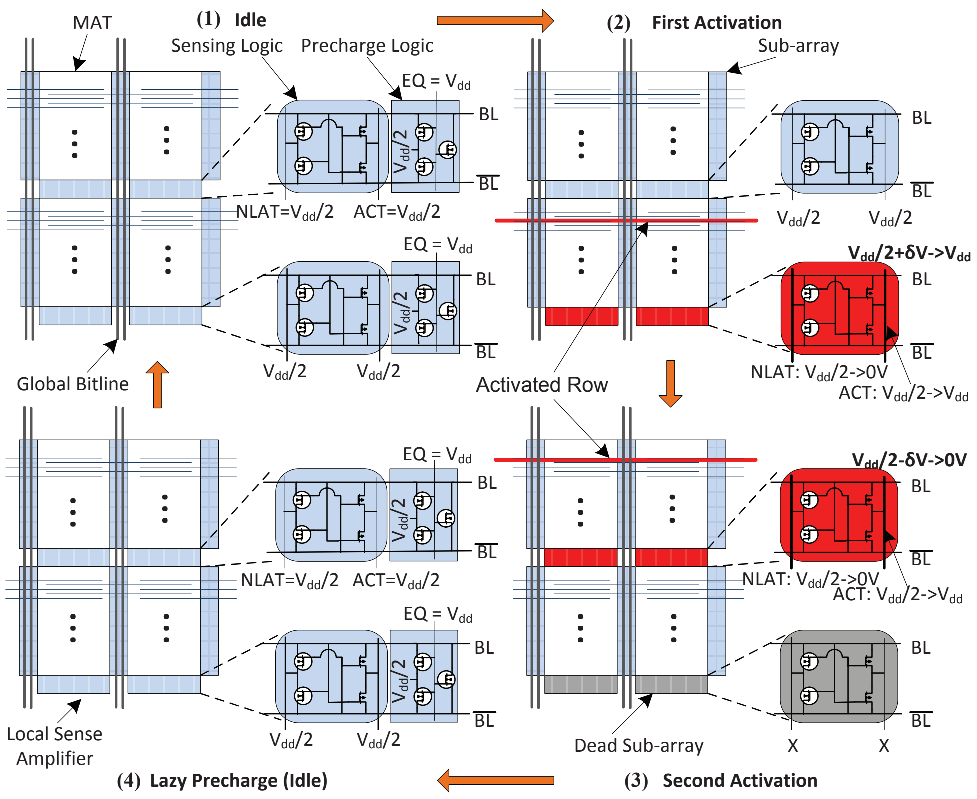


Fig. 2: The circuit operation of activation and precharge

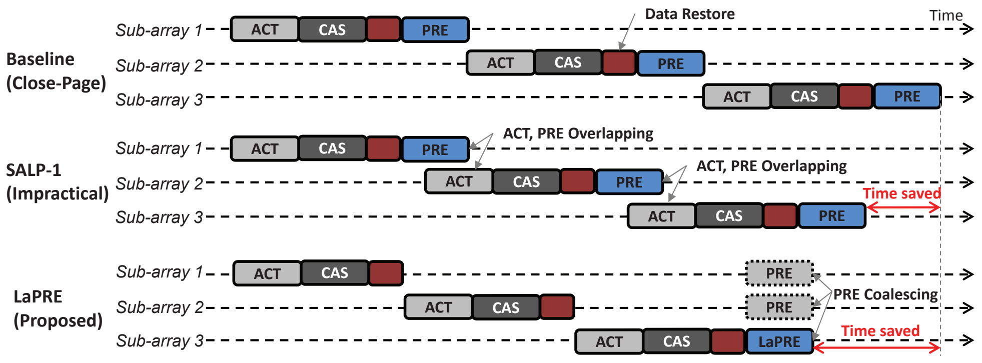


Fig. 3: Timing diagram of the three different memory access schemes

B. DRAM Chip Design

The commodity DRAM chip has provided the infrastructure for LaPRE and thereby LaPRE does not incur any hardware overhead in DRAM chip. As shown in Figure 2(2), once an `ACT` is received, the wordline of the first sub-array is open for data burst. After the data has been restored into the cell, the wordline can be closed without data loss. At this time, the second `ACT` can be issued to open another sub-array without a precharge, which is shown in Figure 2(3). Since only one output of the row address decoder is asserted, the second `ACT` automatically triggers the closing of the old wordline and the opening of a new wordline. Note that no further request can be issued to the first sub-array before a lazy precharge. As a consequence, the row buffer of the first sub-array becomes invisible to MC. We name such a sub-array as a *dead sub-array*.

The aggressive access scheme in LaPRE re-defines the timing constraints on the activation sequence. From the perspective of the first active sub-array, it requires the following `ACT` to be issued after its wordline completely closes. Therefore, the second `ACT` should fulfill the same timing constraint as a `PRE` should obey, which is shown in Table II. In particular, the second `ACT` can only be issued tRTP cycles after a read command `CAS-R`; or, tBURST+tWR cycles after a write command `CAS-W`. Notice that the tRRD and tFAW constraints are still applied to meet the power constraint.

TABLE II: New Timing Constraints on Activation

Prev. Cmd	Curr. Cmd	Timing Constraint	Comments
CAS-R	ACT	tRTP	same bank
CAS-W	ACT	tBURST + tWR	same bank
ACT	ACT	Five-ACT-Window	same rank
I_{ACT}	32.1mA	I_{PRE}	6.2mA

C. DRAM Controller Design

LaPRE should be exposed to the MC so that the MC can make use of the enabled SALP. In particular, *when to precharge a bank* is a critical question to answer when design an MC. In commodity DRAM, Close-Page and Open-Page are two classic row buffer management policies to address the question[3]. In Close-Page, the percharge is initiated immediately after the desired data burst. No benefit of row buffer locality can be exploited. Alternatively, Open-Page policy requires a precharge *if and only if* 1) no request in the queue that addresses the same row (row buffer hit) or four requests have been serviced with row buffer hits; AND 2) there is a request pointing to a different row (row buffer miss). Note that we adopt four row-buffer-hit requests as the threshold to prioritize a row-buffer-miss request based on the results of prior study [2], in which less than five row-buffer-hit requests can be serviced before a precharge. This also takes into account the concern of request starvation and fairness of request scheduling.

Different from the traditional MC, LaPRE-MC has more flexibility to close a bank because multiple sub-arrays can be activated before it issues a lazy precharge. In this work, we propose three memory scheduling schemes to leverage LaPRE for the performance improvement. The three schemes are described as follows.

•**LaPRE-Idle-First**: Similar to the traditional Close-Page policy, LaPRE-MC seeks a memory request in the queue that points to an idle sub-array based on first-come-first-serve (FCFS) policy. A lazy precharge is issued if and only if no such request can be found in the queue.

•**LaPRE-RBH-First**: Similar to Open-Page policy, LaPRE-MC prioritizes the requests that can explore row buffer hit (RBH). A sub-array switching occurs if no such request can be found or the number of requests that have been serviced reaches the threshold to avoid remaining requests' starvation. Finally, a lazy precharge is issued when no request that has row buffer hit or accesses an idle sub-array can be found in the queue.

•**LaPRE-DS-First**: As an extension of LaPRE-RBH-First, LaPRE-MC issues a lazy precharge *once* it detects the head request in the queue points to a dead sub-array (DS). In LaPRE-RBH-First scheme, such head request may be delayed for a long time because LaPRE-MC can continue to drain other requests in the queue. LaPRE-DS-First eliminates the potential delay with more precharges as the payment.

Figure 4 illustrates how the three scheduling schemes work. At the beginning, there are nine requests in the queue (older request has smaller number). The requests having the same color point to the same sub-array and row while the requests with different colors accesses different sub-arrays. For instance, Req-1/3/4/5/6 (light gray blocks) access the same sub-array and same row so that row buffer hit can be exploited. Req-2/7 (dark gray block) or Req-8/9 (black block) access different sub-arrays so that LaPRE can be applied.

Since LaPRE-Idle-First only searches for the request to an idle sub-array, a lazy precharge is needed after the completion of Req-1/2/8. For the same reason, the second lazy precharge is inserted after the completion of Req-3/7/9. Then, each of the remaining requests (Req-4/5/6) needs a precharge because they go to the same sub-array. Compared to the traditional Close-Page policy that needs eight precharges (dashed blue line), LaPRE-Idle-First reduces the number of precharge to four.

When it comes to LaPRE-RBH-First, Req-1/3/4/5 are prioritized due to row buffer hit. Afterwards, Req-2/7 are selected since four requests have been serviced for the previous sub-array. Req-8/9 are then scheduled consecutively. Prior to Req-6, a lazy precharge must be issued to reset the dead sub-array. Finally, Req-6 can be serviced after the precharge. Compared to Open-Page policy, LaPRE-RBH-First reduces the number of precharge from three to one. Also notice that Req-6 is delayed until Req-7/8/9 finish. To avoid the long delay, LaPRE-DS-First issues a precharge once it detects Req-6 comes to the head of queue. After the percharge, the remaining requests are scheduled accordingly. Obviously, LaPRE-DS-First may degrade the performance since it can early terminate a series of requests that are originally featured row buffer hit (e.g., Req-2/7). In addition, LaPRE-DS-First sometimes induces more precharges than LaPRE-RBH-First. For example, another precharge is introduced if the position of Req-6 and Req-8 are swapped in the queue.

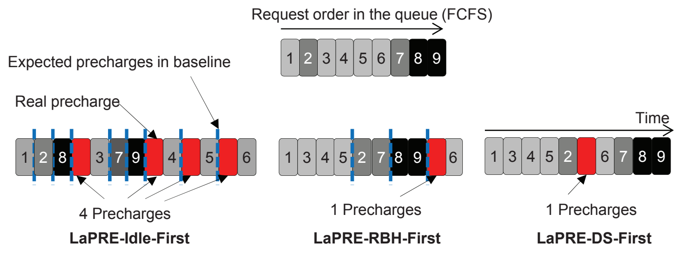


Fig. 4: The proposed memory scheduling schemes

In addition to the row buffer management policy, LaPRE-MC has to account for the power consumed by the simultaneous precharges of multiple sub-arrays. We use the power analysis tool from Rambus [8] to obtain the current dissipation of an activation (I_{ACT}) and a precharge (I_{PRE}), respectively. The result shown in Table II reveals that the precharge current of one sub-array is around 1/5 of the activation current. Therefore, a new Five-ACT-Window constraint is introduced to limit the maximum current of a lazy precharge, where at most five activations can be issued between two precharges.

IV. Evaluation Results

In this work, gem5 [9] (SE mode) is adopted as the simulation platform. We integrate NVMain [10] into gem5, which is a cycle-accurate memory simulator for both DRAM and non-volatile memories. Table III shows the setup of gem5 and NVMain, respectively. All DRAM timing parameters are excerpted from Micron's data sheet [3]. Three memory controllers with the proposed memory scheduling schemes are implemented in NVMain for the evaluation (see Section III-C). Since both Close-Page and Open-Page policies are used in the baseline, these three models are further divided into two classes, where LaPRE-Idle-First is compared to the baseline that employs Close-Page policy while LaPRE-RBH-First and LaPRE-DS-First are compared to the baseline with Open-Page policy.

The selected SPEC2006 CPU benchmark with reference input size [11] and STREAM with all functions [12] are evaluated as multi-programmed testbench. We classify the benchmarks into three categories and select four benchmarks as the representatives for each

[3]Throughout the paper, Close-Page and Open-Page are dedicated to the row buffer management policies used in the commodity DRAM memory controllers. The proposed policies are noted with prefix "LaPRE".

category, which are symbolized as *H*, *M*, and *L* for high (¿10), medium ([1,10]), and low (¡1) miss per kilo instructions (MPKI) of last level cache (LLC). We simply duplicate four copies of each benchmark for the four-core simulation. We run all benchmarks for 500 million instructions for the cache warmup and then the following 100 million instructions for the performance statistics. The weighted instructions-per-cycle (IPC) defined in Eq. 3 is used as the performance criteria throughout the simulation.

$$WeightedSpeedup = \sum_{i=1}^{4} \frac{IPC^{i}_{multi-core}}{IPC^{i}_{standalone}} \tag{3}$$

TABLE III: Simulation Configuration

System Configuration	
Cores	4, ALPHA, out-of-order
CPU Clock Freq.	3 GHz
LDQ/STQ/ROB Size	32 / 32 / 128 entries
Issue/Commit Width	8 / 8
L1-D/L1-I Cache	32kB / 32kB 4-way 2-cycle latency
D-TLB/I-TLB Size	64 / 48 entries
L2 Cache	Shared, Snooping, 4MB, LRU 8-way, 10-cycle latency
Memory	JEDEC-DDR3-1333, 8GB, 64bit channel, 2 ranks, 4Gb chip(×8), 8 banks(64K×8K), tRCD-tCAS-tRP-tWR 10-10-10-10, FR-FCFS, 32-entry queue
Benchmark Classification (MPKI)	

H	STREAM(34.96), mcf(16.26)
	lbm(31.92) gobmk(38.35)

M	bwaves(6.07), milc(5.13), leslie3d(4.30), libquantum(6.95)
L	gamess(0.02), namd(0.12), sphnix3(0.09), soplex(0.3)

A. Performance Analysis

The performance comparison between LaPRE-Idle-First and the traditional Close-Page policy is shown in Figure 6a. All *H* benchmarks and two *L* benchmarks have more than 15% performance improvement. On average 14% performance improvement is observed and the improvement is up to 21% among *H* and *M* benchmarks. In contrast, little performance gain is observed in *L* benchmarks. In particular, `gobmk` and `libquantum` achieve 65.1% and 39% performance gain, respectively. To figure out the source of performance gain, we collect the average number of requests per precharge. Baseline requires one request per precharge due to the Close-Page policy. Alternatively, LaPRE-Idle-First can effectively reduce the overall amount of precharge so that one precharge can service more than one request. For example, `STREAM` has 1.73 requests per precharge while `namd` still has 1 request per precharge. Specifically, 4.19 and 2.43 requests per precharge are observed in `gobmk` and `libquantum`, respectively. The larger request number reflects that more requests share the precharge and thus the precharge overhead is significantly reduced. Even though `soplex` produces 1.43 requests per precharge, the improvement is modest due to low memory intensity.

On the other hand, the performance results of LaPRE-DS-First and LaPRE-RBH-First are shown in Figure 6b. Compared to Figure 6a, LaPRE-DS-First and LaPRE-RBH-First provide less performance improvement, where the overall improvement is only 6.9% (5.7%) for LaPRE-RBH-First (LaPRE-DS-First) and the *H* and *M* benchmarks can obtain 9.8% (6.2%) improvement on average. The reason of less performance gain is two-fold. Firstly, the baseline already leverages Open-Page policy that can service multiple requests for one precharge. Therefore, Open-Page policy effectively offsets the benefit of LaPRE. In addition, the address mapping scheme also affects the effectiveness of LaPRE (see Section IV-B). Therefore, only Close-Page and LaPRE-Idle-First are employed in the following sensitivity study.

B. Sensitivity Study

a) Impact of Number of Sub-arrays: We first sweep the number of sub-arrays in a bank from 1 to 128. Figure 6c shows the results. It is clear that better performance can be obtained if more sub-arrays are employed. It is straightforward that more sub-arrays can reduce the possibility of sub-array conflict that occurs when two requests try to access the same sub-array. LaPRE is always beneficial to `gobmk` as more sub-arrays are deployed. In contrast, the performance improvement is little for `libquantum` beyond 32 sub-arrays. As LaPRE does not induce any extra logics, it can simply leverage the maximum number of sub-arrays to help the performance. Note that this is different from the prior work [5] in which more sub-arrays introduce larger area overhead.

b) Impact of Power Constraint: To examine the impact of power constraint, LaPRE without Five-ACT-Window constraint is simulated and compared to the counterpart where the Five-ACT-Window constraint is applied. As shown in Figure 6d, only `STREAM`, `gobmk` and `libquantum` obtain 4.6%, 8.6% and 6.7% performance improvement, respectively. The additional improvement stems from higher request/precharge rate, which implies these benchmarks fit LaPRE well. The performance gain for other benchmarks, however, are negligible due to little change of request/precharge. As a consequence, the power constraint does not severely degrade the performance.

c) Impact of Address Mapping Scheme: Figure 5 presents the address mapping schemes used in this paper. *Baseline* scheme is applied to Close-Page, Open-Page, LaPRE-RBH-First, and LaPRE-DS-First to maximize the row buffer hit while *LaPRE* scheme is employed in LaPRE-Idle-First to maximize the sub-array level parallelism. To eliminate the interference of rank-/bank-level parallelism, the position of bank and rank segment in baseline scheme is aligned with the bank and rank segment in LaPRE scheme. The baseline address mapping scheme can help explain the less performance gain in Figure 6b. As shown, the least significant bits (LSBs) are assigned to the column bits while the sub-array bits are placed in the relatively higher position. As a result, less sub-array level parallelism can be exploited for the performance improvement.

Among rank-/bank-/sub-array-level parallelism, two experiments are designed to figure out which one is the most important factor for the performance. The rank/bank/sub-array bit segments are alternately placed in the LSB of physical address. The corresponding address mapping schemes are shown in the last three lines of Figure 5, which are labeled SA:RK:BK, RK:BK:SA, and RK:SA:BK. Figure 6e shows the evaluation results. In sum, SA:RK:BK and RK:SA:BK outperforms RK:BK:SA so that bank-level parallelism is most important for the memory performance because banks can completely be accessed in parallel. Furthermore, the sub-array-level parallelism

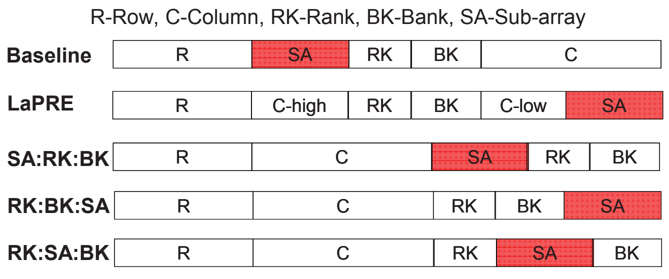


Fig. 5: The address mapping schemes in this work

in LaPRE can at least work as well as rank-level-parallelism when we compare RK:SA:BK with SA:RK:BK and sometimes it even gets better results (`milc` and `leslie3d`). Moreover, under SA:RK:BK scheme, we compare SA:RK:BK to the baseline without LaPRE (labeled as RK:BK). The results show that LaPRE achieves 4.4% and 3.6% performance gain, which indicates LaPRE is still useful even if the sub-array-parallelism is minimized.

To verify the conclusion, we further conduct another experiment by sweeping the number of sub-arrays with RK:BK:SA scheme. As shown in Figure 6f, the result is distinct from Figure 6c: more sub-arrays lead to significant performance degradation. Once more sub-arrays are deployed, the width of sub-array segment in the physical address becomes larger. As a consequence, the bank-level parallelism is weakened due to the left shift of its segments. Even though the performance of `libquantum` is improved as the number of sub-arrays increases from 16 to 64, the overall performance still drops. Therefore, it demonstrates that bank-level parallelism is more important for the performance. More banks, however, requires more peripheral logics and thus reduces area efficiency of the cell and increases the corresponding cost ($/bit). As the clock frequency keeps scaling up, less ranks can be populated in a single channel [13]. Therefore, considering the zero cost, our LaPRE is still meaningful for the performance improvement.

V. Related Works

Several works have been done to leverage the fine-grained memory architecture. In the industry, FCRAM [14] and RLDRAM [15] make use of smaller banks for low-latency memory access. The drawback of these memory modules is the significantly reduced area efficiency, which in turn increases the total cost per bit. On the other hand, fine-grained activation is proposed in [16] so that only a portion of sub-array is activated to reduce activation power. The performance overhead that results from the reduced data bandwidth, however, is not evaluated in this work. Similarly, selective bitline activation (SBA) and single sub-array access (SSA) are proposed in [1]. Nonetheless, the performance results could be misleading because the severe bandwidth reduction and the unaffordable layout overhead are neglected while the problem has been pointed out in [8].

The sub-array level parallelism (SALP) has been explored in [5]. In addition to the SALP-1 that is impractical because of the conflict between activation and precharge in a sub-array, SALP-2 and MASA are also proposed in [5]. SALP-2 prevents the conflict by enabling *selective precharge*. Extra logic is introduced so that the sub-arrays no longer share the same *EQ* signal for the precharge. Instead, each sub-array can have a dedicated *EQ* so that one sub-array can be precharged safely while another sub-array is open. Furthermore, the selective precharge is also applied to MASA with the enhanced isolation logic of the global data path. Compared to LaPRE, both methods requires additional logics and modification to the DRAM chip. As a result, they are supposed to be more costly than LaPRE.

VI. Conclusion

As memory performance becomes the system bottleneck in a CMP architecture, high memory parallelism is mandatory to improve the overall memory performance. In this work, we propose the Lazy Precharge (LaPRE) as a novel memory architecture that can effectively reduce the precharge overhead and thus improve memory parallelism. Compared to the previous works [5] that can cause the conflict of activation and precharge, LaPRE eliminate the conflict by allowing multiple sub-arrays to be activated without the intervention of percharge. A lazy precharge is then shared by all active or dead sub-arrays. In addition, three request scheduling policies are proposed and evaluated to leverage the enhanced memory parallelism. The experimental results show that LaPRE can achieve as much as 65% performance improvement. On average it can obtain 14% and 6.9% performance gain under different scheduling policies. Taking into account the negligible hardware overhead, LaPRE can be favorably adopted by the industry.

References

[1] 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’37*, Jun. 2010, pp. 175–186.

[2] D. Kaseridis, J. Stuecheli, and L. K. John, “Minimalist Open-page: A DRAM Page-mode Scheduling Policy for the Many-core Era,” in *MICRO’44*, Feb. 2011, pp. 24–35.

[3] Micron, “MT41J512M8RA-15E Data Sheet,” http://www.micron.com/products/dram/ddr3-sdram.

[4] JEDEC Solid State Technology Association, “JEDEC Standard: DDR4 SDRAM,” http://www.jedec.org/sites/default/files/docs/JESD79-4.pdf, Sep. 2012.

[5] Y. Kim, V. Seshadri, D. Lee, J. Liu, and O. Mutlu, “A Case for Exploiting Subarray-Level Parallelism (SALP) in DRAM,” in *ISCA’39*, Jun. 2012, pp. 368–379.

[6] D. Lee, Y. Kim, V. Seshadri, J. Liu, and O. Mutlu, “Tiered-Latency DRAM: A Low Latency and Low Cost DRAM Architecture,” in *HPCA’19*, Feb. 2013, pp. 615–626.

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

[8] T. Vogelsang, “Understanding the Energy Consumption of Dynamic Random Access Memories,” in *MICRO’43*, Dec. 2010, pp. 363–374.

[9] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi *et al.*, “The gem5 Simulator,” *Computer Architecture News*, vol. 39, no. 2, pp. 1–7, Aug. 2011.

[10] M. Poremba and Y. Xie, “NVMain: An Architectural-Level Main Memory Simulator for Emerging Non-volatile Memories,” in *ISVLSI’12*, Aug. 2012, pp. 392–397. [Online]. Available: http://www.nvmain.org

[11] Standard Performance Evaluation Corporation, “SPEC2006 CPU,” http://www.spec.org/cpu2006.

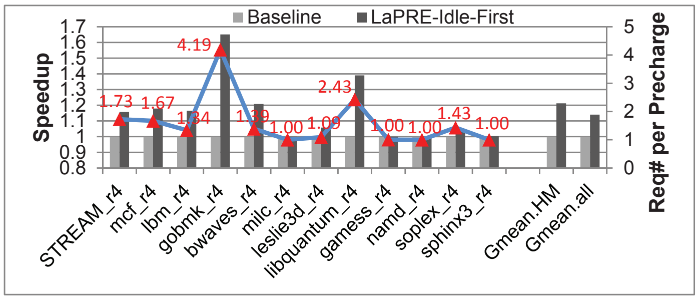
[12] J. D. McCalpin, “STREAM Benchmark,” http://www.cs.virginia.edu/stream.

[13] B. Jacob, S. W. NG, and D. T. Wang, *Memory Systems: Cache, DRAM, Disk*. Morgan Kaufmann, 2007.

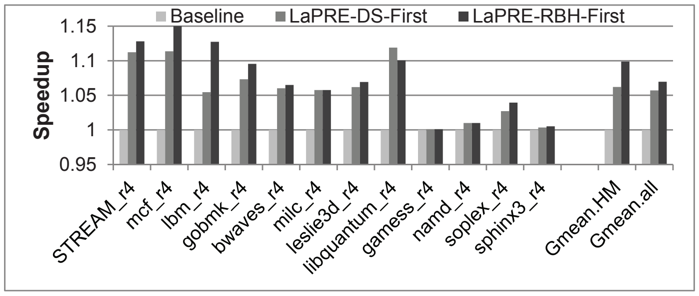
[14] Fujitsu, “Memory Consumer FCRAM 512M Bit MB81EDS516445,” 2010.

[15] Micron, “MT44K64M18RCT-125 Data Sheet,” http://www.micron.com/products/dram/rldram-memory.

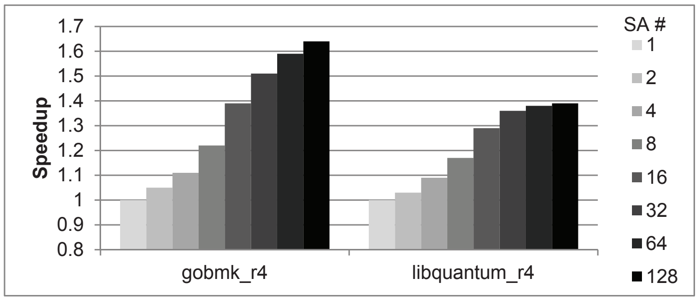
[16] E. Cooper-Balis and B. Jacob, “Fine-Grained Activation for Power Reduction in DRAM,” *MICRO*, vol. 30, no. 3, pp. 34–47, 2010.



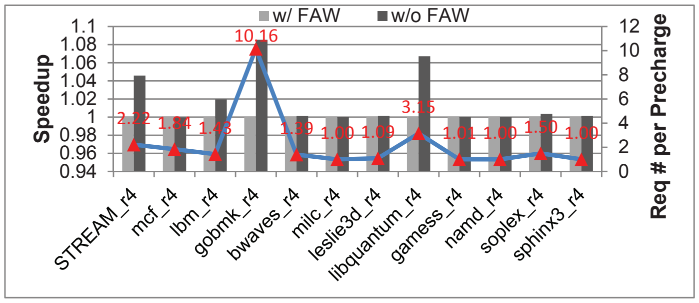
(a) Class-1



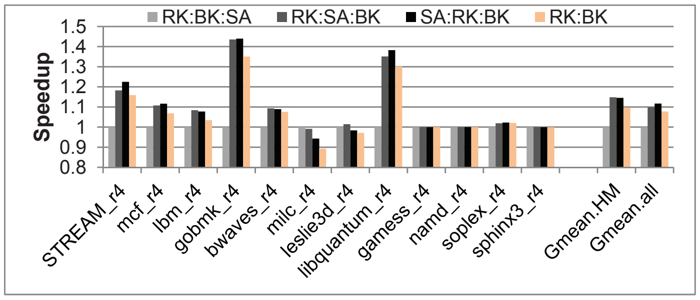
(b) Class-2



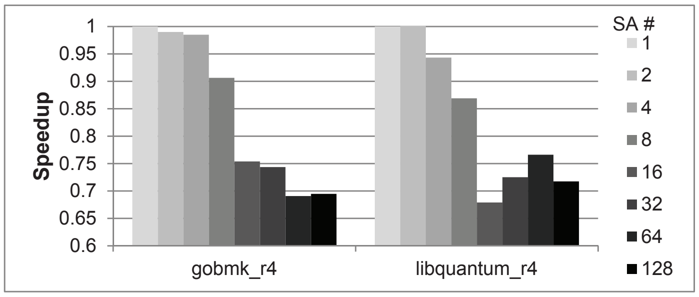
(c) Impact of Number of Sub-arrays



(d) Impact of Power Constraint



(e) Impact of Address Mapping



(f) Evidence of the importance of bank-level parallelism

Fig. 6: Performance results. (a) Class-1: LaPRE-Idle-First vs. Close-Page; (b) Class-2: LaPRE-DS-First and LaPRE-RBH-First vs. Open-Page; (c) Impact of Number of Sub-arrays; (d) Impact of Power Constraint; (e) Impact of Address Mapping; (f) Evidence of the importance of bank-level parallelism.