NUAT: A Non-Uniform Access Time Memory Controller

Wongyu Shin Jeongmin Yang Jungwhan Choi Lee-Sup Kim
Department of Electrical Engineering
Korea Advanced Institute of Science and Technology (KAIST)
{wgshin, jeongmin, cjwdream}@mvlsi.kaist.ac.kr, leesup@kaist.ac.kr

Abstract

With rapid development of micro-processors, off-chip memory access becomes a system bottleneck. DRAM, a main memory in most computers, has concentrated only on capacity and bandwidth for decades to achieve high performance computing. However, DRAM access latency should also be considered to keep the development trend in multi-core era. Therefore, we propose NUAT which is a new memory controller focusing on reducing memory access latency without any modification of the existing DRAM structure. We only exploit DRAM's intrinsic phenomenon: electric charge variation in DRAM cell capacitors. Given the cost-sensitive DRAM market, it is a big advantage in terms of actual implementation.

NUAT gives a score to every memory access request and the request with the highest score obtains a priority. For scoring, we introduce two new concepts: Partitioned Bank Rotation (PBR) and PBR Page Mode (PPM). First, PBR is a mechanism that draws information of access speed from refresh timing and position; the request which has faster access speed gains higher score. Second, PPM selects a better page mode between open- and close-page modes based on the information from PBR. Evaluations show that NUAT decreases memory access latency significantly for various environments.

1. Introduction

Modern computer systems cannot operate without memory systems. To achieve high performance computers, memory systems should consider many factors such as capacity, bandwidth, and access latency. However, Dynamic Random Access Memory (DRAM), a standard main memory in most computing systems, has been developed focusing only on capacity and bandwidth. This trend was maintained with the help of cache systems which are essential in today's computers to hide long DRAM access latency. Even though the cache systems were successful in compensating for the speed difference between CPU and DRAM, they are reaching the limit with the advent of multi-core systems [29, 9]. Concurrent memory requests from diverse programs, running on a multi-core processor simultaneously, induce the lack of spatial locality [26] which is a fundamental principle of cache systems.

The fundamental solution to overcome the limit that cache systems are facing is to reduce DRAM access latency. In other words, DRAM systems should take into account what we used to believe cache systems can do. If DRAM access latency becomes lowered, memory systems can be configured much more efficiently than ever before. This paper, therefore, concentrates on reducing DRAM access latency.

Some previous works [24, 15] were successful in reducing latency. However, the area required by these works makes significant additional DRAM manufacturing cost. To minimize this unwanted cost increment, several proposals exploit DRAM's inner structure. These proposals understand DRAM's physical operation, and make use of the performance slack found in DRAM circuits. Specifically, subarray structure, usually not exposed to memory controllers, is exploited with small additional circuits in DRAM [11]. A long bit-line is split into two shorter segments by an isolation transistor and a low-latency segment is utilized as a cache [13]. These proposals are so brilliant, but still require a modification of the conventional DRAM structure though it is a minor modification. Even small changes of DRAM give rise to unexpected phenomena concerning signal integrity, power integrity, layout, and so on [19, 3, 14]. The area overhead to solve these phenomena makes the implementation hard.

Our final objective is to design a new memory controller focusing on reducing DRAM access latency without any modification of the existing DRAM structure. *NUAT* is a new memory controller that can achieve the final objective. NUAT gives a score to every memory request and a priority is given to the request which gains the highest score (see Sec. 4 for details). A scoring system of NUAT basically makes it possible to construct an adaptive memory control system according to a designer's intention, but the first version of NUAT introduced in this paper concentrates on our final objective.

The observation exploited by this work is that charge stored in DRAM cell capacitors is not fixed, but decreases from the last refresh time to the next refresh time. The amount of charge stored in DRAM cell capacitors determines an initial voltage difference (ΔV) of sense amplifiers. Sense amplifiers are circuits for sensing ΔV, and the sensing speed is strongly affected by the initial ΔV (see Sec. 2.3 for details). Through this observation, we arrived at a key insight: "The DRAM row access latency is a function of the elapsed time from when the row was last refreshed." This paper explains the key insight and proposes a memory controller (NUAT) that exploits it cost-effectively.

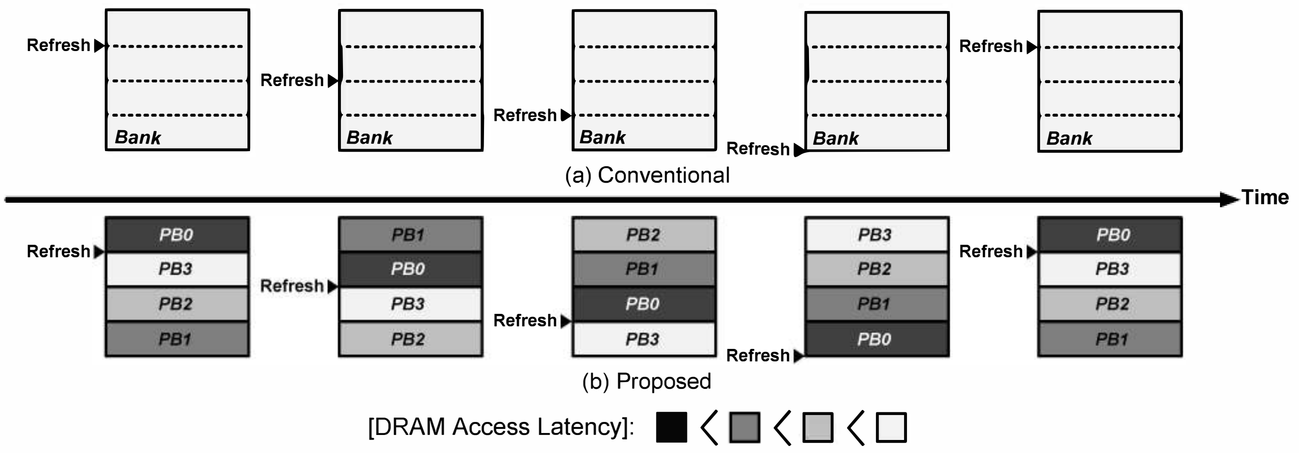


Figure 1: Partitioned Bank Rotation (PBR): DRAM bank state transition

In this work, two new concepts are introduced to support NUAT. The one is *Partitioned Bank Rotation (PBR)* and the other is *PBR Page Mode (PPM)*. First, PBR determines PB# based on refresh position and timing. *Partitioned Bank (PB)*, a part of a DRAM bank, contains information about access speed; PB0 is the fastest part of the bank and PB3 is the slowest part of the bank. PB# is rotated with time as shown in Fig. 1 because refresh is performed periodically for data retention. Second, PPM is a mechanism that selects a better page mode between open- and close-page modes. The optimal page mode is determined from row-buffer hit-rate and DRAM timing parameters [6]. Since PB# has information about access timing parameters, PPM is determined from PB#.

This work makes the following contributions.

- A new memory controller, NUAT, is proposed based on the observation that charge in DRAM cell-capacitors changes periodically. To our knowledge, this is the first work finding performance slack in charge variation of cell-capacitors.
- NUAT reduces memory access latency without any modification of the existing DRAM structure. This will be a considerable advantage for actual implementation.
- PBR and PPM are proposed to support NUAT. PBR decides PB# which contains access speed information and PPM is determined based on PB#.
- We evaluate our new memory controller quantitatively. Circuit simulation is performed using 55nm DRAM technology which is publicly available. We also do system simulations in variety of configuration using various workloads.

2. Background

2.1. DRAM Structure

Fig. 2 shows a structural hierarchy of DRAM. Some previous works [11, 13] exploit inner bank structures. We, however, observed an inner bank phenomenon which is related to sense amplifiers and cells. The details of this phenomenon are explained in Sec. 3.

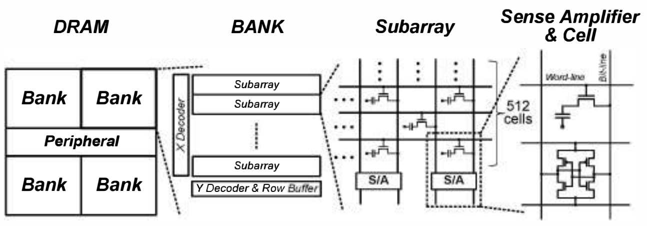


Figure 2: DRAM Structure Hierarchy

2.2. DRAM Operation

DRAM performs three operations: read, write, and refresh. To complete each operation, a series of DRAM commands should be issued with pre-defined time intervals. Fig. 3 describes the command sequences with the timing parameters for each operation. The timing parameters concerning our work are *tRCD*[1], *tRAS*[2], and *tRC*[3] which are all relevant to row activation as depicted in Fig. 3. Row activation is a

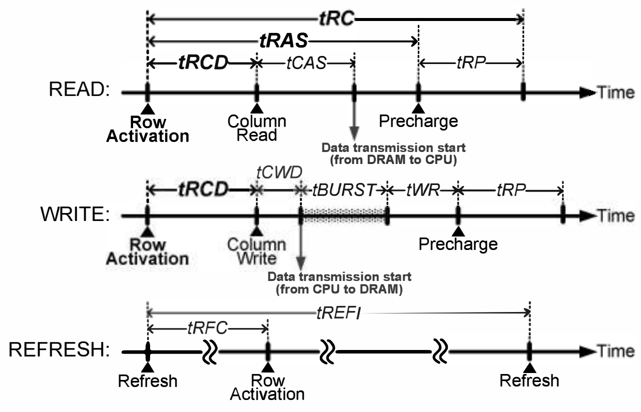


Figure 3: Command Sequences with Timing Parameters

[1] *tRCD*: Row to Column command Delay [6]
[2] *tRAS*: Row Access Strobe [6]
[3] *tRC*: Row Cycle [6]

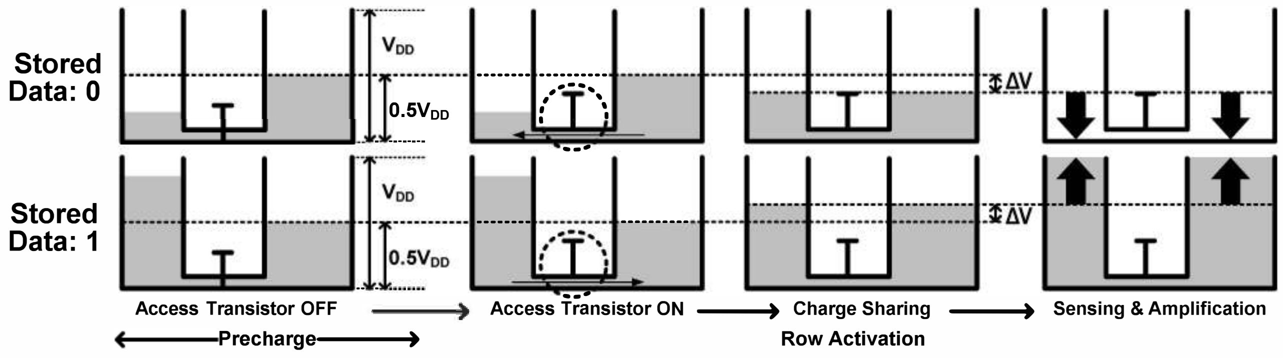


Figure 4: Precharge and Row Activation: a voltage difference incurred by charge sharing is amplified by sense amplifiers.

process which is driven by sense amplifiers, the key components of our observation. These three timing parameters will be optimized by our proposed memory controller, NUAT. The details will be explained from Sec. 4.

2.3. Row Activation

In the previous sections (2.1 & 2.2), we explained what components we observed and what timing parameters will be optimized. The components are sense amplifiers and the timing parameters are *tRCD*, *tRAS*, and *tRC*. This section provides background about how the timing parameters are optimized based on the phenomenon in the sense amplifiers. Row activation, driven by the sense amplifiers, is illustrated in this section. This is a key to understanding our observation which is discussed in Sec. 3.

- **Analogy Model of a DRAM Cell.** A DRAM cell consists of one capacitor and one access transistor. If we think of electrical charge as water and think of a capacitor as a water tank, an analogy model can be constructed as in Fig. 5. This analogy model will be used to explain our observation.

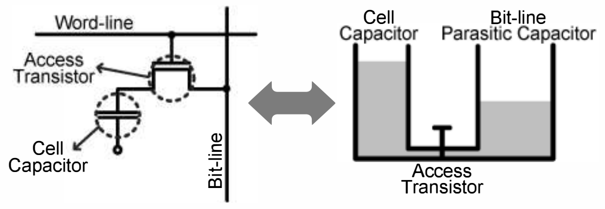


Figure 5: DRAM Cell structure & Analogy model

- **Row Activation Process.** The process is illustrated in Fig. 4 using the analogy model (depicted in Fig. 5). Before row activation, a bank should be precharged. Physically, precharge is a process that set the voltage of bit-lines as $0.5V_{DD}$ (reference voltage). When a row activation command is issued, an access transistor is turned on, and charge sharing occurs. If stored data is '0', charge is moved from a bit-line to a cell capacitor by the voltage difference. If, on the other hand, stored data is '1', charge is moved from a cell capacitor to a bit-line. When charge sharing is finished, ΔV is created, and this ΔV is amplified by a sense amplifier.

3. Motivation

What we observed is charge variation in cell capacitors and its effect on sense amplifiers. As described in Sec. 2.3, charge stored in a cell capacitor makes ΔV which is a seed voltage difference for a sense amplifier. ΔV has an effect on the response time of the sense amplifier (it is shown in Sec. 4.1 with circuit simulation.). Since ΔV is created by the charge stored in the cell capacitor, the speed of the sense amplifier is also affected by the stored charge. Consequently, the speed of a sense amplifier is affected by charge variation in a cell capacitor. However, DRAM timing parameters have been determined assuming that the sense amplifier is driven by the smallest ΔV which can be created by charge sharing.

The more important observation is that the stored charge in the cell capacitor varies periodically, and therefore the sensing speed of the sense amplifiers also changes periodically as described in Fig. 6. The periodicity is made by refresh operation because refresh is the process that fills the charge, which is leaked out with time, into the cell periodically. The sensing speed is fastest when the row activation is performed right after refresh. On the other hand, the sensing speed is the slowest when the row activation is performed right before refresh. Through this observation, we arrived at a key insight that composes a basic principle of NUAT: "The DRAM row access latency is a function of the elapsed time from when the row was last refreshed."

In this paper, we exploit this observation due to three main reasons which are as follows:

- **The Impact on Latency.** According to [25], the biggest portion of access time is occupied by sensing and restoring process which are relevant to charge sharing and sense amplifier operation. It means that the bottleneck of reducing latency exists in charge sharing and sense amplifier operation.
- **Periodicity.** Periodicity always gives a big chance to controllers. Usually, a lot of gadgets are required to obtain information needed for optimization. For instance, temperature sensors are required for temperature information for dynamic thermal management (DTM) [5, 8, 12]. Designing temperature sensors is burdensome to

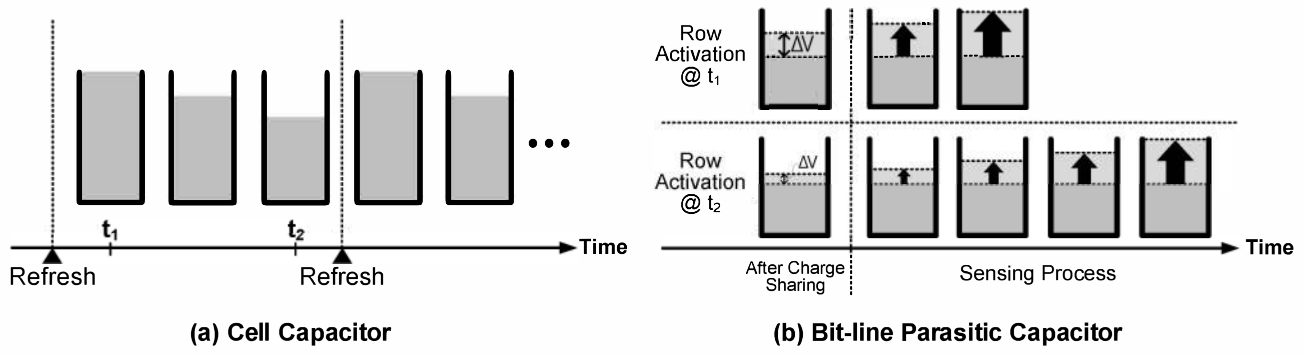


Figure 6: (a) Periodicity: Stored charge in a cell capacitor varies periodically with refresh process. (b) Speed variation: A sense amplifier has different sensing speed according to the amount of charge in a cell capacitor.

VLSI designers and requires much additional cost. However, periodicity provides us with free information because the information can be made from calculation using the periodicity. DRAM has an intrinsic periodicity for data retention, and we can draw information from the periodicity. Specifically, row access latency information can be estimated from the timing and position of refresh operation as explained before. Finally, this periodicity leads to cost-effective optimization.

- **Area Overhead.** Area overhead is so important in the cost-sensitive DRAM market. Thanks to intrinsic periodicity of DRAM, any additional equipment is not required in DRAM. The only required area overhead is some computational blocks in memory controller (in CPU) to draw information from the periodicity.

4. Overview of NUAT

A non-uniform access time memory controller consists of three parts: *PBR Acquisition Block*, *PPM Decision Maker*, and *NUAT Table* as described in Fig. 7. NUAT basically adopts a scoring system; the request which acquires the highest score is first processed. NUAT Table is a set of scoring criteria by which all memory requests are scored. It is supported by PBR acquisition block and PPM decision maker. Each part is explained below.

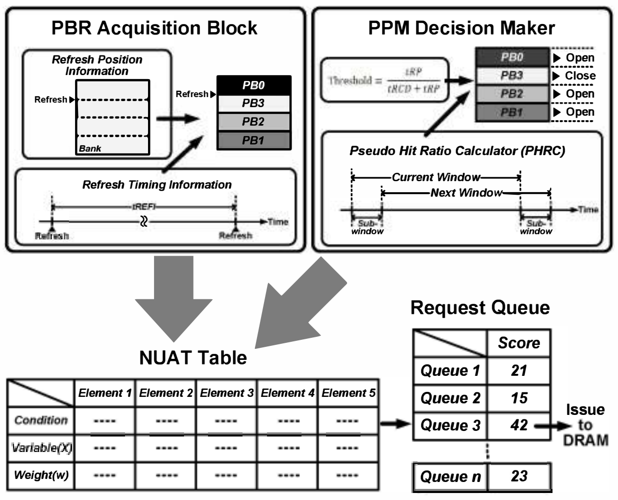


Figure 7: Overall Configuration of NUAT

- **PBR Acquisition Block.** Partitioned Bank Rotation (PBR) is a mechanism to find PB number (PB#) which contains information of access speed. The concept of PBR was illustrated in Fig. 1. Every cell should be refreshed in 64ms and the refresh is performed row by row with the interval of *tREFI*[4]. Using the refresh position and timing, we can calculate PB#. Since refreshing 8 rows at once with the interval of 8×*tREFI* is common [17], PB# is updated every 8×*tREFI*. In **Sec. 5**, it is described in detail.

- **PPM Decision Maker.** PBR Page Mode (PPM) is a mechanism that selects a better page mode between open- and close-page modes. There are some previous works about page mode policy [10]. PPM, however, determines the page mode based on the PB# derived by PBR acquisition block. Since the optimal page mode is determined by timing parameters (*tRCD & tRP*) [6] and PB# has the information of *tRCD*, NUAT should change the page mode depending on PB# for optimization. PPM decision maker performs this optimization. In **Sec. 6**, it is described in detail.

- **NUAT Table.** A NUAT table provides a memory controller with decision criteria for scoring (see Table.1). The score is a sum of five sub-scores which are called *Element Score (ES)* (ES1~ES5). Each ES is calculated by three fields: condition, variable, and weight. The NUAT table was configured to accomplish our goal, reducing access latency. In **Sec. 7**, it is described in detail.

5. Partitioned Bank Rotation (PBR)

A NUAT memory controller should first calculate *PB#* by *Partitioned Bank Rotation (PBR)* acquisition block because PPM and NUAT Table are based on PB#. Thus, we explain PBR in this section before PPM and NUAT Table are described. First, a key concept of PBR is introduced and the actual design of PBR acquisition block is illustrated with latency analysis.

[4] *tREFI*: Refresh Interval period [16]

5.1. PBR Acquisition

A key concept of PBR acquisition is finding a relative address of a request-row-address (RRA) to the last-refresh-row-address (LRRA). The relative address implies the elapsed time from when the row was last refreshed. Specifically, since refresh is performed row by row with time interval *tREFI* and it is repeated with the period of 64ms (DRAM retention time), the relative address to the LRRA represents the degree of charge in cell capacitors, which leads to sensing speed difference. (In this paper, we assume that a linear counter is used for refresh for simplicity.) Therefore, PB#, which contains information about access speed, can be decided by a relative address of a RRA to the LRRA. Fig. 8 describes the concept of PBR acquisition.

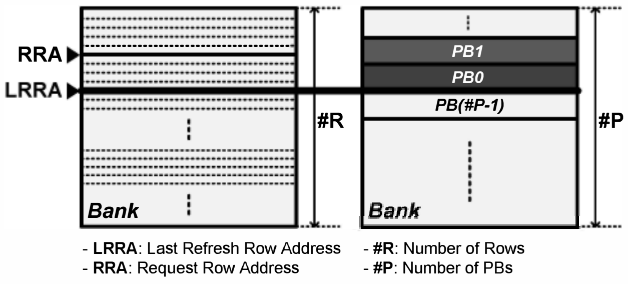


Figure 8: A Concept of PBR Acquisition

A relative position should be divided by the number of rows in one PB to find PB# because PB is a group of multiple rows. For a simple calculation in a memory controller, bitwise-shift-right operation is used. (1) is the equation to draw PB#.

$$PB\# = (LRRA - RRA) \gg (\log_2 \#R - \log_2 \#P) \qquad (1)$$

It means MSB (log2 #P) bits of difference between LRRA and RRA (relative address of RRA to LRRA). #P is the number of PBs and (log2 #P) is the minimum bits that can represent #P. #P can be decided by memory controller designers; memory controllers can have more chances for optimization as #P increases, but area and power overhead in a memory controller also increases. The sensitivity of the number of PBs is evaluated in Sec. 8.

5.2. Latency Analysis

For actual implementation of PBR, we need to verify access time sensitivity according to charge variation in cell capacitors. The sensitivity occurs mainly from sense amplifiers. Thus, in this section, sense amplifier's sensitivity to an initial voltage difference (ΔV) is verified through circuit evaluation using SPICE. Publicly available 55nm DDR3 2Gb process technology [28, 21] is used for simulation parameters such as bit-line and cell capacitance. Simulation environment is scaled to meet DRAM timing parameters such as *tRCD* and *tRAS*.

Fig. 9 (a) shows sensitivity of a sense amplifier. Initial voltage difference (ΔV) is swept from the voltage when a cell capacitor is fully charged (right after refresh) to the voltage when the cell capacitor should be refreshed (right

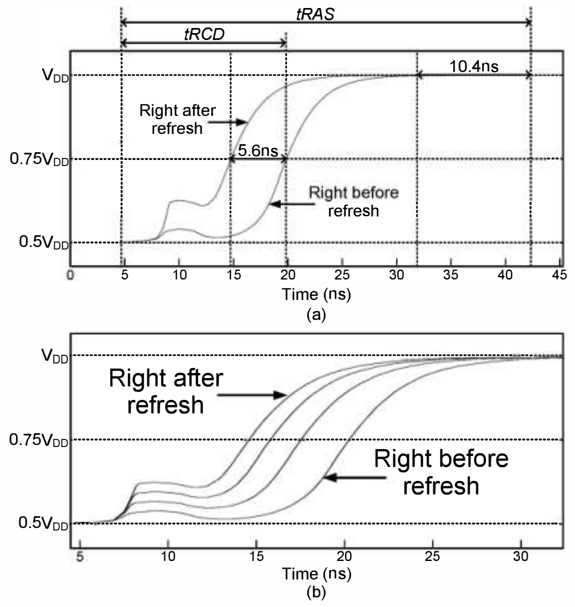


Figure 9: Sensitivity of Sense Amplifiers (Circuit Evaluation)

before refresh). It is verified that sense amplifiers are sensitive depending on ΔV through the circuit evaluation. *tRCD* can be reduced by 5.6ns and *tRAS* can be reduced by 10.4ns. Assuming memory controller operates at 800MHz, *tRCD* / *tRAS* can be lowered by up to 4 cycles / 8 cycles depending on situation.

Fig. 9 (b) shows a nonlinearity of a sense amplifier. Nonlinearity is induced from various factors in electrical circuits [4]. This nonlinearity can be removed with additional assistant electrical circuits [20, 7]. However, we cannot change the circuits because one of our final objectives is not to modify the existing DRAM structure as mentioned before. To take account of this non-linearity, PBR needs a modification, and this is illustrated in Sec. 5.3.

5.3. Modified PBR Acquisition

PBR acquisition block should be modified to cover nonlinearity of sense amplifiers. A key idea to take nonlinearity into account is that PB size should be non-uniform; the number of rows in each PB should be different. Fig. 10 is an example in the case of four PBs.

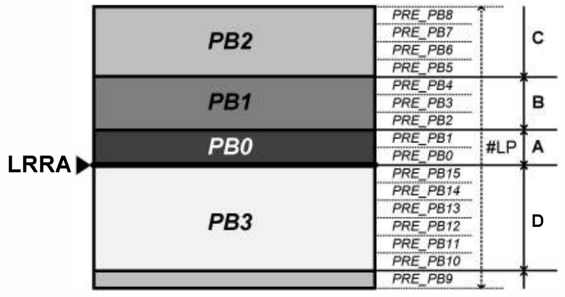


Figure 10: Modified PBR Acquisition

Equation (1) should also be modified. A modified method has two steps: linear division and non-linear grouping. First, linear division is the same as (1). In Fig. 10, the dotted lines represent the linear division and we call the result of linear division PRE_PB. (2) is the equation for the first step.

$$PRE\_PB\# = (LRRA - RRA) \gg (\log_2 \#R - \log_2 \#LP) \quad (2)$$

#LP represents the number of linear PBs. Memory controller designers can set #LP depending on situation. The second step is non-linear grouping. Each size of PB# is decided based on the result of a circuit simulation. According to the size, the final PB# is determined as follows:

$$\begin{array}{llll} \textit{if} & 0 \leq PRE\_PB\# < A, & PB\# = 0 \\ \textit{else if} & A \leq PRE\_PB\# < B, & PB\# = 1 \\ \textit{else if} & B \leq PRE\_PB\# < C, & PB\# = 2 \\ \textit{else} & PB\# = 3 & \end{array}$$

This is an example of four PBs for explanation. Specific design parameters and values are shown in Sec. 8.

6. PBR Page Mode (PPM)

PBR Page Mode (PPM) is a mechanism finding an optimal page mode of PBs. PPM finds the optimal page mode of each PB from the information generated by PBR and *Pseudo Hit-Rate Calculator (PHRC)*. PBR can draw PB# which contains access time information and PHRC approximates current hit-rate of row-buffer.

6.1. Pseudo Hit-Rate Calculator (PHRC)

PHRC calculate current row-buffer hit-rate. Row-buffer hit-rate is calculated as follows:

$$Hit\_Rate = \frac{\#Column\_Acess - \#Row\_Activation}{\#Column\_Acess} \quad (3)$$

#Column_Access is the number of column access commands and #Row_Activation is the number of row activation commands. If Column_Access and Row_Activation commands for read are only counted, it will be a read-hit-rate.

For an exact calculation of current hit-rate, every command issued during window should be recorded (assuming that window size is big enough). In other words, every command history during window should be stored in a memory controller, which requires a lot of storage area. PHRC, however, does not record every command in order to minimize area overhead. PHRC only stores command history during subwindow (see Fig. 11). Specifically, every command issued during sub-window B is recorded, but history of sub-window A is assumed as the average of history of current window.

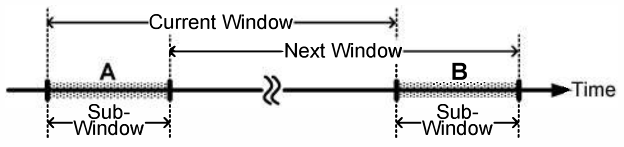


Figure 11: Sub-Window for Hit-Rate Approximation

To calculate this, *Window Ratio* should be first set as follows:

$$Window\_Ratio = \frac{Window\_Size}{Sub\_Window\_Size} \quad (4)$$

Using Window Ratio and the number of commands during current window (#Current_Window), we estimate the number of commands during A (#A) as follows:

$$\#A = \frac{\#Current\_Window}{Window\_Ratio} \quad (5)$$

After the estimation of the number of commands during subwindow A, #A is displaced by #B as follows:

$$\#Next\_Window = \#Current\_Window + (\#B - \#A) \quad (6)$$

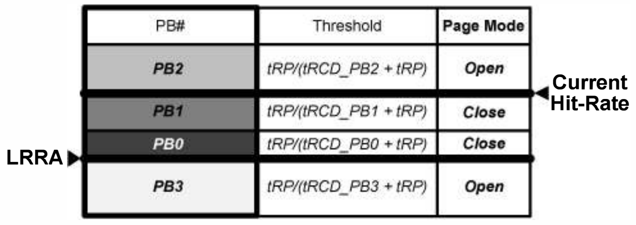
The estimated number (for both column access and row activation) is used again in (3), and pseudo-hit-rate is finally calculated. Specific values such as Window Ratio are shown in Sec. 8.

6.2. PPM Decision Maker

PPM decision maker compares current hit-rate calculated by PHRC with *Threshold* of PBs. Threshold is a boundary hit-rate between open- and close-page modes. Threshold is determined by equation (7) [6].

$$Threshold = \frac{tRP}{tRCD + tRP} \quad (7)$$

If current hit-rate is bigger than Threshold, open-page mode is better than close-page mode. On the other hand, if current hit-rate is less than Threshold, close-page mode is better than open-page mode. However, all PBs have different threshold values because *tRCD* is decided according to PB#. Therefore, a better page mode is determined depending on PB#. This is illustrated in Fig. 12 and specific values are shown in Sec. 8.



PB#	Threshold	Page Mode
PB2	tRP/(tRCD_PB2 + tRP)	Open
PB1	tRP/(tRCD_PB1 + tRP)	Close
PB0	tRP/(tRCD_PB0 + tRP)	Close
PB3	tRP/(tRCD_PB3 + tRP)	Open

Current Hit-Rate
LRRA

Figure 12: Threshold & Page Mode of PBs

7. NUAT Table

NUAT Table is first developed for optimization of DRAM operation. NUAT Table can basically be configured for various purposes by modulating weight values (see Table 1). The first version of NUAT table, however, focuses on reducing latency to accomplish our final objective. In this section, score calculation is first explained, and the modeling of elements and the decision of weights are following.

Table1: NUAT Table

- *ACT*: Activation command - *WQ_L*: Write Queue Length - *#D*: # of PBs
- *COL*: Column command - *HW*: High Watermark of WQ - *WC*: Wait Cycle
- *PRE*: Precharge command - *LW*: Low Watermark of WQ

Element	*Element 1: OPERATION-TYPE*					
Condition	*WQ_L < LW*		*LW < WQ_L < HW*		*WQ_L > HW*	
	Read	*Write*	*Read*	*Write*	*Read*	*Write*
Variable(X)	1	0	Previous Variable		0	1
Weight(w)	w1					

Element	*Element 2: WAIT*			*Element 3: HIT*			
Condition	*ACT*	*COL*	*PRE*	*ACT*	*COL*		*PRE*
					Read	*Write*	
Variable(X)	WC		0	0	2	1	0
Weight(w)	w2			w3			

Element	*Element 4: PB*			*Element 5: BOUNDARY*			
Condition	*ACT*	*COL*	*PRE*	*ACT*		*COL*	*PRE*
				PB# : the last	*PB# : others*		
Variable(X)	#D−PB#	0	0	−1	+1	0	0
Weight(w)	w4			w5			

7.1. Score Calculation

Each element has its own score called *Element Score (ES)* and the final score is the sum of these five ESs as follows:

$$Score = ES1 + ES2 + ES3 + ES4 + ES5 \qquad (8)$$

Each ES is decided by multiplying a variable (x) by a weight (w), and the final score is expressed as follows:

$$Score = \sum_{k=1}^{5} w(k) \cdot x(k) \qquad (9)$$

Specific explanation about the variables (x) and weights (w) is described in Sec 7.2 and 7.3.

7.2. Modeling of Elements

The variables are determined according to the conditions for their own reasons. They are explained in this section.

- **Element 1: OPERATION-TYPE.** Conventionally, a memory controller gives a priority to read operation and a write queue is managed using Low Watermark (LW) and High Watermark (HW) of the write queue. NUAT also adopts this concept in NUAT table. For a scoring system of NUAT, we built a hysteresis model as illustrated in Fig. 13. ES of OPERATION-TYPE is determined according to the write queue length and the kind of path. ① is a draining path and ② is a filling path of the write queue (see Fig. 13).

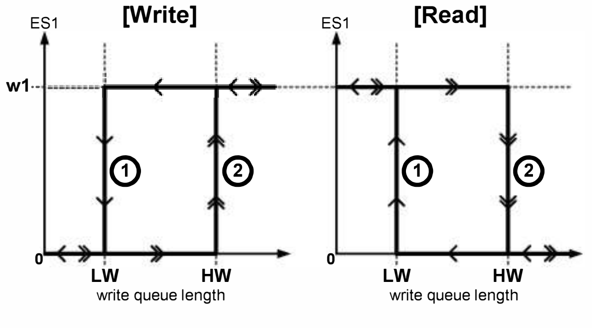


Figure 13: Hysteresis Model of OPERATION-TYPE

- **Element 2: WAIT.** WAIT is for the entering order in a request queue. The variable is the number of wait cycles in the queue. If the condition of Element 2 has core (or thread) information and w2 is appropriate, the NUAT table can be configured focusing on fairness. In the NUAT table we designed, it will be the same as a *FCFS*[5] scheduler if only w1 and w2 are considered (w3,w4,w5=0).
- **Element 3: HIT.** HIT is for hit-rate. If a request has the same row address as an already opened row, ES of HIT can get a score; a read request obtains 2×w3 and a write request obtains 1×w3. The reason of different variables is explained in Sec. 7.3. If only w1, w2, and w3 are considered (w4,w5=0), it will be the same as a *FR-FCFS*[6] scheduler.
- **Element 4: PB.** This element considers PB#. The key concept is that a request which can be processed faster should have a priority because PB# is changed with time. Since PB# is about activation process, only activation commands are taken into account in creating ES of PB.
- **Element 5: BOUNDARY.** Boundaries between two PBs can be a warning zone or a promising zone as depicted in Fig. 14. Warning zone is a region where next PB# is bigger than current PB# and promising zone is vice versa. If an activation command is in a transition region (a region where PB# changes after the next refresh operation), ES of BOUNDARY is 1 or -1 depending on the type of zone; 1 is for warning zone and -1 is for promising zone.

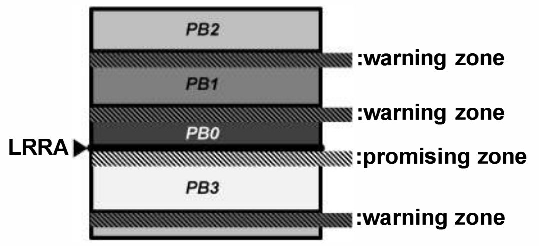


Figure 14: Warning & Promising Zone of Boundaries

7.3. Decision of Weights

Weights are one of the design fields that designers can decide. There are a lot of combinations of weights. Every combination has their own characteristics, and designers can choose one of them. For our goal (reducing latency), we first make a priority order: OPERATION-TYPE (w1) ≥ HIT (w3) > PB (w4) > BOUNDARY (w5) > WAIT (w2). The more priority an element has, the bigger weight value is set. Fig. 15 describes the scale of weights and ES variation scope. Specific explanations are as follows.

- **OPERATION-TYPE (w1) ≥ HIT (w3).** On path ② (see Fig. 13), any read command has more scores than

[5] *FCFS*: First Come First Served
[6] *FR-FCFS*: First Ready First Come First Served [23,22]

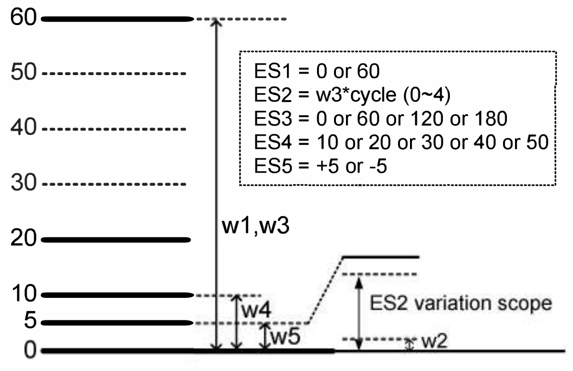


Figure 15: Weights & ES Variation Scope

write command. On the other hand, on path ①, a column read command which is hit with a row activated for write has the same score as a write column command hit with the row. This can increase hit-rate and reduce read latency because write-to-read turnaround time is much less than row activation time. To model this case in NUAT, w1 should be the same as w3, and variables of read and write for Element 3 should be different. The situation is described in Fig. 16.

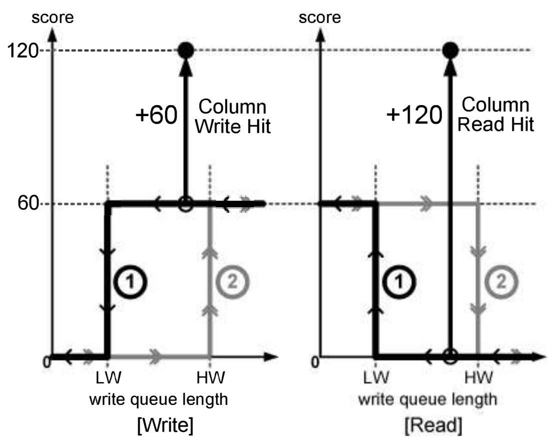


Figure 16: The Reason for Different Variables in Element 3

- **HIT (w3) > PB (w4).** HIT (Element 3) has more priority than PB (Element 4) to maximize hit-rate. To give a priority to Element 3, the weight is set as described in Fig. 15. Since the maximum of ES4 is 50 (less than w3), it is impossible that ES4 changes the priority set by ES1 and ES3.
- **PB (w4) > BOUNDARY (w5).** An activation command in a warning zone obtains +5 to give more priority than activation commands in the same PB. On the other hand, an activation command in a promising zone obtains -5 to give less priority than activation commands in the same PB. The scope of ES5 is from -5 to +5, and therefore it cannot change the priority set by ES1, ES3, and ES4.
- **BOUNDARY (w5) > WAIT (w2).** In this system, WAIT has the least priority. ES2 cannot change the priority determined by ES1, ES3, ES4, and ES5 because the scope of ES2 is from 0 to 4.

8. Evaluation

For simulations, we developed a DDR3-SDRAM simulator based on a memory system simulator, USIMM [16, 27] which is used for MSC (Memory Scheduling Championship [2]). We modified it to evaluate our work. The workloads we used for the evaluation are described in Table 2, which are used for MSC. To perform multi-core evaluation, workload combinations which are randomly selected are used. We generated 32 combinations for the 2-core evaluation and also made 32 combinations for the 4-core evaluation.

Table 2: Workloads

COMMERCIAL	comm1, comm2, comm3, comm4, comm5
SPEC	leslie, libq
PARSEC	Black, face, ferret, fluid, freq, stream, swapt, MT-canneal, MT-fluid
BIOBENCH	mummer. tigr

A basic system configuration is shown in Table 3. A processor model follows the USIMM default model. In a memory controller, we modeled a read queue, a write queue, and address mapping as in Table 3. The DRAM model we used consists of 1 channel, 1rank/channel, 8 banks/rank. One bank has 8K rows and 1K columns and cache line size is 64 bytes. Timing parameters are from a DDR3 SDRAM data sheet [1].

Table 3: Basic System Configuration

Processor	- Clock Frequency: 3.2GHz - ROB Size: 128 - Retire Width: 2 - Fetch Width: 4 - Pipeline Depth: 10
Memory Controller	- Memory Bus Frequency: 800MHz - Read Queue Capacity: 64 - Write Queue Capacity: 64 - Write Queue High Watermark: 40 - Write Queue Low Watermark: 20 - Address Mapping: Open-Page Baseline mapping [6]
DRAM	- 1 Channel, 1 Rank, 8 Bank, 8K Rows, 1K Columns, 64 Bytes Cache line size - tRCD: 15ns, tRAS: 37.5ns, tRC: 52.5ns [1]

Our evaluation focuses on memory access latency. The main evaluation we must do is the effect of PB on memory access latency. Therefore, sensitivity of the number of PBs is evaluated. Through the circuit simulations (see Sec. 5.2.), we decided that the maximum number of PBs is 5 because 5.6ns (see Fig. 9) is 5 cycles in 800MHz (Memory bus frequency). We set #LP (The number of linear PBs) as 32 and each PB has a different number of PRE_PBs due to non-linearity of sense amplifiers. The PB configuration from 2PB to 5PB is depicted in Fig. 17.

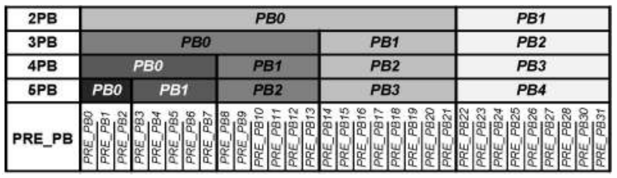
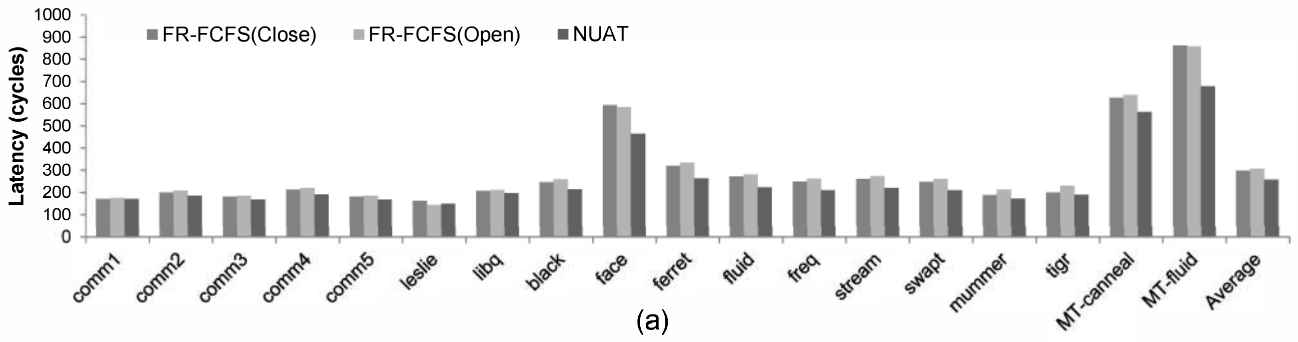


Figure 17: PB Configuration



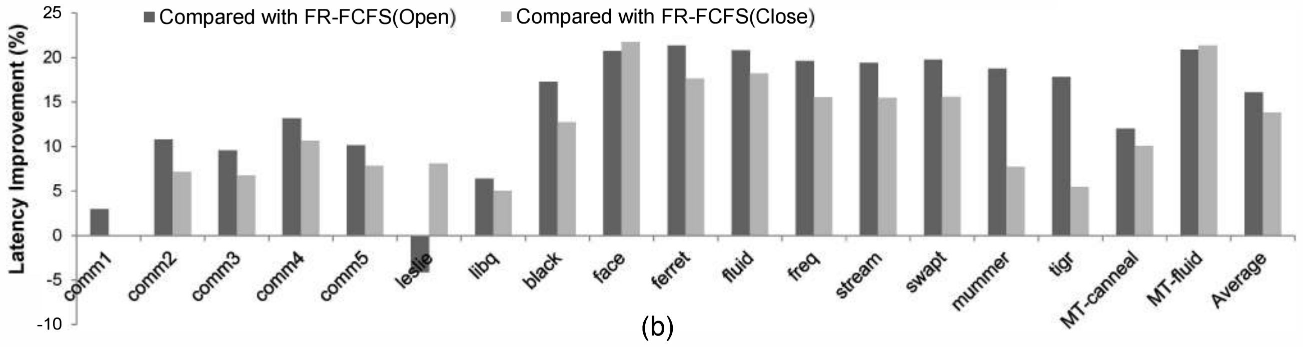


Figure 18: Read Access Latency: (a) Access Latency of NUAT, FR-FCFS(Open), and FR-FCFS(Close) (b) Percentage of reducing latency compared with FR-FCFS open- and close-page mode.

NUAT parameters are shown in Table 4. Timing parameters of each PB are specifically illustrated. Sub-window size is 1024, and it needs 1K bits (128 bytes) additional storage space in a memory controller (in CPU). Weights in NUAT Table are decided based on the principle explained in Sec. 7.3.

Table 4: NUAT Parameters

PBR	- #P: 5 - #LP: 32 - PB0: PRE_PB0 ~ PRE_PB2 (3 PRE_PBs) - PB1: PRE_PB3 ~ PRE_PB7 (5 PRE_PBs) - PB2: PRE_PB8 ~ PRE_PB13 (6 PRE_PBs) - PB3: PRE_PB14 ~ PRE_PB21 (8 PRE_PBs) - PB4: PRE_PB22 ~ PRE_PB31 (10 PRE_PBs) - [PB0] tRCD_0: 8, tRAS_0: 22, tRC:34 (unit: cycle) - [PB1] tRCD_1: 9, tRAS_1: 24, tRC:36 (unit: cycle) - [PB2] tRCD_2: 10, tRAS_2: 26, tRC:38 (unit: cycle) - [PB3] tRCD_3: 11, tRAS_3: 28, tRC:40 (unit: cycle) - [PB4] tRCD_4: 12, tRAS_4: 30, tRC:42 (unit: cycle)
PPM	- Sub-Window Size: 1024 (unit: cycle) - Window Ratio: 256
NUAT Table	- [Weight] w1=60, w2=0.0001, w3=60, w4=10, w5=5

To evaluate performance, we compare NUAT with FR-FCFS. Open- and close-page modes are both taken into account for the evaluation of the PPM decision maker. It is reasonable to compare with FR-FCFS because our key idea is involved in Element 4 and Element 5 in NUAT Table (see Table 1). As explained in Sec. 7, if only Element 1 and Element 2 are considered, it will be the same as FR-FCFS.

9. Results

9.1. Impact on Latency

Fig. 18 (a) shows the read latency of various workloads and Fig. 18 (b) compares NUAT with FR-FCFS open- and close-page mode in percentage. This simulation is performed based on 5PB configuration. As expected, NUAT reduces the memory access latency significantly for most workloads. The access latency is reduced up to 21.3% in both cases of Ferret (open) and MT-fluid (close). On average, NUAT can reduce the latency by 16.1% and by 13.8% when compared with FR-FCFS open- and close-page mode. However, there are two workloads that degrade the latency. The access latency is increased by 4.1% in comparing with open-page mode for Leslie and 0.07% in comparing with close-page mode for Comm1. The analysis is as follows.

- **Hit-Rate Difference between Two Page Modes** This is the main reason of the Leslie case. Read-page hit-rate is 0.65/0.28 for FR-FCFS open- and close-page mode. The difference between for FR-FCFS open- and close-page

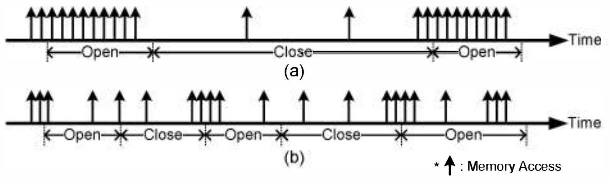


Figure 19: Analysis for the Leslie case

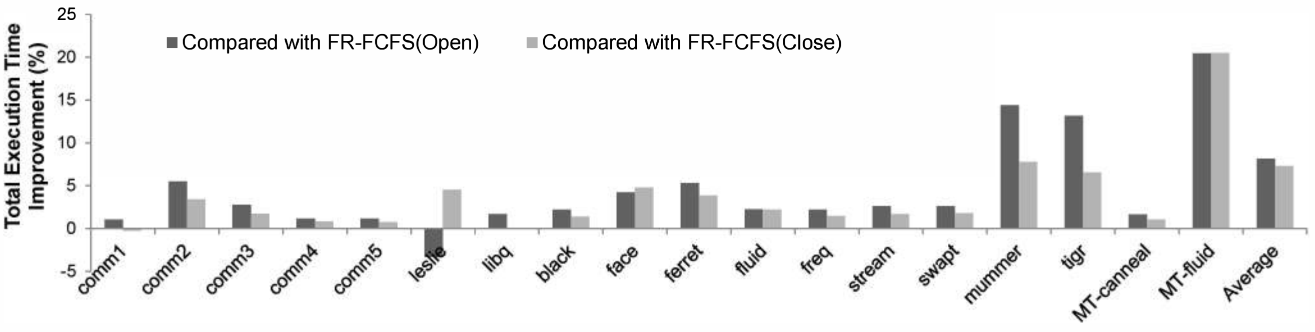


Figure 20: Total Execution Time

mode is 0.36 which is the biggest among the workloads (The average hit-rate difference is 0.08). It means that memory access occurs frequently, but not in a bursting situation. Specifically, since PHRC needs tracking time to calculate hit-rate, if memory access occurs like Fig. 19 (b), performance can be degraded. If memory access occurs like Fig. 19 (a), PHRC can track the favorable page-mode well. Consequently, reduced latency is overwhelmed by the side-effect of PHRC for the Leslie case.

- **Probabilistic Problem.** This is the main reason of the Comm1 case. Since PB is rotated with time according to refresh timing and position, and every PB has their own size due to non-linearity of sense amplifiers, this makes probabilistic situation. Specifically, when one memory access request is issued, the probability that the row address is in PB0 / PB1 / PB2 / PB3 / PB4 is $\frac{3}{32}$ / $\frac{5}{32}$ / $\frac{6}{32}$ / $\frac{8}{32}$ / $\frac{10}{32}$ (see Fig. 17). The percentage of PB3 and PB4 among the row accesses is 80% for Comm1. This is the highest among the workloads (average: 59%). Surely, there should be no degradation with the only probabilistic issue. However, some workloads have a negative effect from PHRC. Usually, since the negative effect is a little (if existing), it can be covered by reduced latency, but Comm1 is unfortunate. Leslie has too much of the side-effect, so this is not the unfortunate case (only 13% access to PB4 for Leslie).

9.2. Total Execution Time

Fig. 20 shows total execution time improvement in percentage. NUAT (in 5PB configuration) is compared with FR-FCFS open- and close-page mode. The total execution time is reduced up to 20.4% in both cases of MT-fluid. On average, NUAT can reduce the total execution time by 8.1% and by 7.3% when compared with FR-FCFS open- and close-page mode. Due to the parallelism such as out-of-order execution, equipped in processing cores, the total execution time improvement is smaller than the latency improvement in average. However, each workload has a different relationship between the latency improvement and the total execution time improvement depending on the characteristic of the workloads. For example, the improvement rate in the latency and in the total execution time is almost same for MT-fluid. It means MT-fluid is a data-intensive workload, and therefore its performance heavily depends on memory access latency. On the other hand, the difference between the improvement rate in the latency and in the total execution time is quite big for Fluid because it has a lot of instructions that can hide long DRAM access latency. Consequently, NUAT can have more advantages in computing systems for data-intensive workloads such as data centers.

9.3. The Number of PBs and Hardware Overhead

Fig. 21 shows sensitivity to the number of PBs (#PB). The baseline is a 2PB configuration and the reduced cycles compared with the 2PB configuration are depicted. As expected, the reduced cycles increase with the number of PBs. The rate of performance improvement follows the non-linearity characteristic of sense amplifiers. Specifically, as the number of PBs is increased, the rate of improving performance is decreased. The sensitivity is more distinct as the number of cores increases. However, the hardware overhead is increased too. If we choose a 4PB configuration, we can lower hardware overhead because a 5PB configuration needs 3 bits to store PB information. In other words, the 5PB configuration requires one more bit than the 4PB configuration for each entry of a request queue, resulting in 128 bits more storage area in a memory controller (in CPU) (we assume 64 read and write queue entries.). The area for 128 bits is not critical in the memory controller. Therefore, our main simulation adopted the 5PB configuration as described in Sec. 9.1 and 9.2.

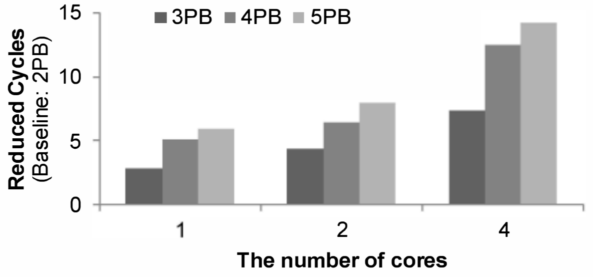


Figure 21: Sensitivity to the number of PBs

9.4. Multi-Core Effects

Fig. 22 shows the improvement of total execution time in percentage depending on the number of cores. The simulation is conducted with 5PB-configuration NUAT, and the improvement is compared with two page modes: open- and close-page mode. For 1-/2-/4-core systems, NUAT can reduce the total execution time by 4.8%/6.2%/21.9% in average compared with FR-FCFS open-page mode. When compared to FR-FCFS close-page mode, the total execution time is reduced by 3%/7.2%/20.9% in average for 1-/2-/4-core systems. It is obviously shown that NUAT improves performance much more with the number of cores increased on computer systems. This is because spatial locality, which is a fundamental principle of cache systems, diminishes with diverse programs running simultaneously on a multi-core environment [29, 9]. In addition, when taking into account the trend of system integration such as ’CPU+GPU’, it is highly expected that NUAT can have more impact than described in Fig. 22 because the system should share one DRAM, not different DRAMs like DDR3 for CPU and GDDR5 for GPU.

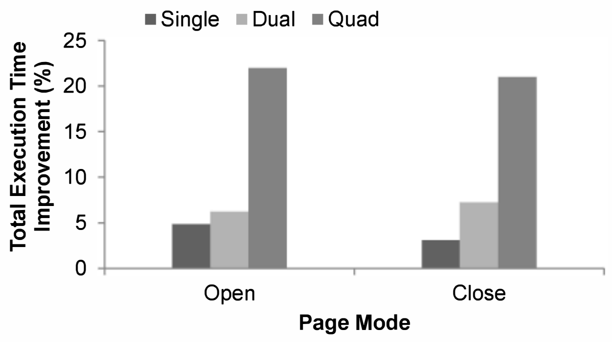


Figure 22: Multi-Core Effects

10. Discussion

It is obviously true that sense amplifiers have sensitivity to initial voltage difference which depends on remaining charge in cell capacitors, and the amount of charge can be estimated by refresh timing and position. We exploited this apparent observation for designing a memory controller in this paper. However, it is also obvious that there are concerns about process-voltage-temperature (PVT) variations that should be considered for implementation. Thus, it is meaningful to discuss about PVT-variation, and therefore we suggest some methods that can address this issue in this section.

10.1. Binning

PVT-variation is unavoidable in modern VLSI technology, and it is evident that it becomes more serious with process scaling. If it is impossible to design a system that is totally independent of PVT-variation, exploiting it could be a method. The method is binning which is an already essential process for manufacturing high-end VLSI chips. NUAT can provide a methodology for binning. The methodology is based on two facts: 1) the "worst-case" has determined the performance of VLSI chips; 2) the "worst-case" is so rare. It means DRAM has large margin "in average" for normal operation. From this observation, DRAMs can be assorted into multiple #PB-DRAMs: 1PB-DRAM~5PB-DRAM as depicted in Fig. 23. The more margin a DRAM device has, the more #PB (the number of PBs) memory controllers can consider. In other words, PVT variations can be hidden in the binning process of DRAM. DRAM vendors make a profit by selling DRAMs with lower latency by higher price instead of selling only DRAMs targeted for the worst-case, that is, 1PB-DRAMs. It should be noted that stricter binning is possible if DRAM has more margins. Specifically, 5PB is not a definite number; it depends on how much margins the DRAM has.

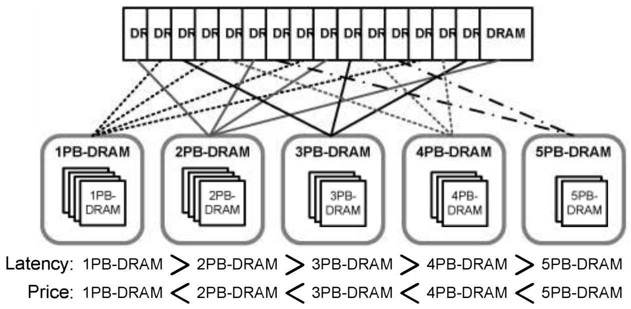

DRAM
1PB-DRAM
2PB-DRAM
3PB-DRAM
4PB-DRAM
5PB-DRAM
Latency: 1PB-DRAM > 2PB-DRAM > 3PB-DRAM > 4PB-DRAM > 5PB-DRAM
Price: 1PB-DRAM < 2PB-DRAM < 3PB-DRAM < 4PB-DRAM < 5PB-DRAM

Figure 23: Binning Process for NUAT

10.2. Architectural Support

By adopting stricter binning, DRAM manufacturing companies can make a profit while processor companies increase their system performance. However, "perfect binning" is not a simple process, and therefore architectural support should be followed for implementation of NUAT. The architectural support is based on the observation: "faulty words are too rare in DRAM, and almost all faulty words only have one faulty cell [18]" It means that if architectural support like ECC can be equipped, binning process does not have to be perfect. For example, even if a DRAM device, which should be 4PB-DRAM due to a few words that have 1-bit faulty cell, is assorted into 5PB-DRAM, this device can operate normally as 5PB-DRAM with ECC that can correct 1-bit. If stronger ECC is equipped in DRAM, binning process can be released as much. Not only ECC but also many other architectural supports can be considered. We believe that architectural support for NUAT will be a meaningful research area in the future.

11. Conclusion

Reducing DRAM access latency is getting more important for continuous development of computers in the architecture world. However, on the other hand, it is one of the hardest things in the VLSI implementation world due to the cost-sensitive DRAM market. Therefore, it is time to consider both aspects to overcome this problematic situation. To make both computer architects and DRAM manufacturers satisfied, we proposed a new memory controller, NUAT.

The speed variation induced by charge variation in cell capacitors is characterized as PB# by PBR. A PPM decision maker generates an optimized page mode for each PB. Finally, NUAT Table makes a final decision. NUAT adopts a scoring system to take many factors into account. The first version of NUAT Table introduced in this paper focuses on reducing latency. NUAT can reduce the latency by 15% and improve total execution time by 7.7% in average without any modification of the existing DRAM structure. We believe that NUAT will be a dominant control scheme in the future because of implementation-inspired optimization and the expandability.

Acknowledgment

The authors would like to thank Youngsuk Moon of SK Hynix for discussions and feedback. This work was funded by SK Hynix.

References

[1] SK Hynix DDR3 SDRAM Data Sheet. [Online]. Available: http://www.skhynix.com/products/computing/computing.jsp?info.ramCategory=computing&info.ramKind=19&info.eol=NOT&posMap=computingDDR3

[2] (2012) MSC: Memory Scheduling Championship. [Online]. Available: http://www.cs.utah.edu/~rajeev/jwac12/

[3] Z. Al-Ars, S. Hamdioui, A. J. Van de Goor, and S. Al-Harbi, "Influence of Bit-Line Coupling and Twisting on the Faulty Behavior of DRAMs," *TCAD*, 2006.

[4] Behzad Razavi, Design of Analog CMOS Integrated Circuits, international edition ed. McGraw-Hill, 2001.

[5] D. Brooks and M. Martonosi, "Dynamic Thermal Management for High-Performance Microprocessors," in *HPCA*, 2001.

[6] Bruce Jacob, Spencer W. Ng, and David T. Wang, Memory Systems(Cache, DRAM, Disk), 1st ed. Morgan Kaufmann, 2008.

[7] C. Cui, T.-S. Kim, S.-K. Kim, J.-K. Cho, S.-T. Kim, and B.-S. Kim, "Effects of The Nonlinearity of The Common-Gate Stage on The Linearity of CMOS Cascode Low Noise Amplifier," in *RFIC*, 2011.

[8] Y. Ge and Q. Qiu, "Dynamic Thermal Management for Multimedia Applications Using Machine Learning," in *DAC*, 2011.

[9] M. K. Jeong, D. H. Yoon, D. Sunwoo, M. Sullivan, I. Lee, and M. Erez, "Balancing DRAM Locality and Parallelism in Shared Memory CMP Systems," in *HPCA*, 2012.

[10] 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.

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

[12] A. Kumar, L. Shang, L.-S. Peh, and N. Jha, "System-Level Dynamic Thermal Management for High-Performance Microprocessors," *TCAD*, 2008.

[13] D. Lee, Y. Kim, V. Seshadri, J. Liu, L. Subramanian, and O. Mutlu, "Tiered-Latency DRAM: A Low Latency and Low Cost DRAM Architecture," in *HPCA*, 2013.

[14] Y. Li, H. Schneider, F. Schnabel, R. Thewes, and D. Schmitt-Landsiedel, "DRAM Yield Analysis and Optimization by a Statistical Design Approach," *TCAS-I*, 2011.

[15] Micron Technology Inc., RLDRAM3 Datasheet, 2011.

[16] N. Chatterjee, R. Balasubramonian, M. Shevgoor, S. Pugsley, A. Udipi, A. Shafiee, K. Sudan, M. Awasthi and Z. Chishti, "USIMM: the Utah Simulated Memory Module". University of Utah, Technical Report, 2012.

[17] P. Nair, C.-C. Chou, and M. K. Qureshi, "A Case for Refresh Pausing in DRAM Memory Systems," in *HPCA*, 2013.

[18] P. J. Nair, D.-H. Kim, and M. K. Qureshi, "ArchShield: Architectural Framework for Assisting DRAM Scaling by Tolerating High Error Rates," in *ISCA*, 2013.

[19] Y. Nakagome, M. Aoki, S. Ikenaga, M. Horiguchi, S. Kimura, Y. Kawamoto, and B. Kiyoo Itoh, "The Impact of Data-Line Interference Noise on DRAM Scaling," *JSSC*, 1988.

[20] J. Radic, A. Djugova, and M. Videnovic-Misic, "Linearity Issue in 2.4GHz 0.35um BiCMOS Low Noise Amplifier," in *TELSIKS*, 2009.

[21] Rambus. (2010) DRAM Power Model. [Online]. Available: http://www.rambus.com/us/downloads/document_abstracts/products/dram_power_model_disclaimer.html

[22] S. Rixner, "Memory Controller Optimizations for Web Servers," in *MICRO*, 2004.

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

[24] Y. Sato, T. Suzuki, T. Aikawa, S. Fujioka, W. Fujieda, H. Kobayashi, H. Ikeda, T. Nagasawa, A. Funyu, Y. Fuji, K. Kawasaki, M. Yamazaki, and M. Taguchi, "Fast Cycle RAM (FCRAM); A 20-ns Random Row Access, Pipe-Lined Operating DRAM," in *VLSI*, 1998.

[25] Y. H. Son, O. Seongil, Y. Ro, J.W. Lee, and J. H. Ahn, "Reducing Memory Access Latency with Asymmetric DRAM Bank Organizations," in *ISCA*, 2013.

[26] K. Sudan, N. Chatterjee, D. Nellans, M. Awasthi, R. Balasubramonian, and A. Davis, "Micro-Pages: Increasing DRAM Efficiency with Locality-Aware Data Placement," in *ASPLOS*, 2010.

[27] UTAH ARCH. (2012) USIMM: the Utah Simulated Memory Module. [Online]. Available: http://utaharch.blogspot.kr/2012/02/usimm.html

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

[29] Q. Zhao, D. Koh, S. Raza, D. Bruening, W.-F. Wong, and S. Amarasinghe, "Dynamic Cache Contention Detection in Multi-threaded Applications," in *VEE*, 2011.