CDAR-DRAM: An *In-situ* Charge Detection and Adaptive Data Restoration DRAM Architecture for Performance and Energy Efficiency Improvement

Chuxiong Lin[1], Weifeng He[1]*, Yanan Sun[1], Zhigang Mao[1], Mingoo Seok[2]
[1]Dept. of Micro-Nano Electronic, Shanghai Jiao Tong University, Shanghai, China
[2]Dept. of Electrical Engineering, Columbia University, New York, NY, USA
*Corresponding author: hewf@sjtu.edu.cn

2021 58th ACM/IEEE Design Automation Conference (DAC) | 978-1-6654-3274-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/DAC18074.2021.9586146

***Abstract*—As the capacity of DRAM continues to grow, the refresh operation rapidly becomes the performance and power-efficiency bottleneck. Also, restore time, the time given for recharging cells post access, makes an increasingly large amount of negative impact on performance. To tackle these problems, in this paper, we propose an *in-situ* charge detection and adaptive data restoration DRAM (CDAR-DRAM) architecture, which can dynamically adjust the refresh rate and also relax the constraints on restore time. The proposed CDAR-DRAM employs a low-cost skewed-inverter-based detector, which can reduce the excessive timing margins that prior work added to guarantee the functionality of leaky DRAM cells under the worst-case temperature condition. Moreover, an adaptive DRAM refresh and restore scheme is proposed, which can switch automatically between two modes: (i) a refresh mode that supports adaptive refresh rate, and (ii) a restore mode that relaxes the constraints on restore time dynamically for cells having sufficient charge. With the transistor- and architecture-level simulations, we evaluate the CDAR-DRAM in an 8-core system across different workloads. Compared with the prior art, the proposed architecture achieves a 9.4% improvement in system performance and a 14.3% reduction in energy consumption, without requiring the time-consuming profiling process which many prior works employed.**

Keywords—DRAM, refresh, restore, row activate time, in-situ charge detection, adaptive restore time, multi-rate refresh

I. Introduction

In recent years, the refresh and restore operations of dynamic random access memory (DRAM) have drawn great attention due to their increasing impact on the performance of DRAM and computer systems. As the DRAM capacity continues to grow, the refresh overhead keeps increasing. For 64GB DRAM chips that are under development, the refresh overhead is estimated to degrade throughput by up to 50% and account for 50% of the total power consumption of DRAM [1]. Also, the restore time, i.e., the time that is given to recharge a DRAM bitcell after the read or write access, continues to dominate DRAM access latency in past decades [2-5]. The impact of restore time even becomes worse for the technology node of 20 nm or below [6, 7]. Consequently, mitigating the refresh overhead and speeding up the bitcell restoration process are paramount to successfully develop the next-generation high-performance and energy-efficient DRAM.

Prior works tried to reduce the refresh overhead in DRAM by exploiting the large spread of data retention time t_{RET} of bitcells [1, 8-11]. Most of bitcells indeed can retain their data much longer than the pessimistically-set refresh window t_{REFW} of 64ms [1, 12]. Therefore, prior works proposed to perform infrequent refresh operations for the majority of the cells that are not leaky whereas frequent refresh operations only for a small number of leaky cells. To identify those leaky cells, prior works relied on profiling the t_{RET} for every row of bitcell array. However, the profiling procedure would take hours to even days to collect all the minimum t_{RET}. This is because t_{RET} depends on data pattern and also varies over time due to the variable retention time phenomenon [13, 14].

On the other hand, prior works tried to shorten the restore time (and refresh time) by leveraging the fact that the restoration time can be smaller if a bitcell still retains a large amount of charge on its capacitor or it is about to refresh [3-6, 15-19]. To do so, prior works proposed to add extra isolation transistors to the bitlines in DRAM chips to reduce the charge loss of bitcells [3, 19]. Some other works instead, exploited the different charge levels of bitcells due to the temporal correlation of operations [4-6, 15-18]. For instance, the restore truncation (RT) scheme [6] aimed to restore a cell's voltage just enough to sustain data until the next refresh operation. Fig. 1 shows an example case. If a "read" operation is issued during the time window *win0*, the RT scheme initiates a normal restore process as the next refresh is distant in time; during *win1*, it can restore the cell's voltage only up to V_1 since the subsequent refresh operation will recharge the cell to V_{DD}. Prior works estimated the required time to restore/refresh cells to a partial voltage (e.g., V_1) via profiling and simulation, and modulated DRAM timing parameters such as refresh cycle time (t_{RFC}), row activate time (t_{RAS}), and row address to column address delay (t_{RCD}) for controlling the restoration (or refresh) strength/duration.

However, most of the prior works have relied on profiling and simulation under the *static worst-case condition* and seldom account for dynamic variations such as temperature. Indeed, the actual working temperature of DRAM is very likely to be less than the worst-case temperature (85 ºC) [16]. Such an approach, therefore, makes DRAM operate with a large amount of unexploited margin during most of its runtime. To confirm this, we simulated the lower bounds of t_{RAS} from 45 ºC to 85 ºC and found that t_{RAS} can be set 31% smaller at 65 ºC and 45% smaller at 45 ºC as compared to 85 ºC (Fig. 2).

In this work, we aim to replace the static profiling and simulation-based approach with *in-situ* sensing hardware for the goal of exploiting those pessimistically-set margins in runtime. To do so, we propose an *in-situ charge detection and adaptive data restore DRAM* (CDAR-DRAM) architecture. It consists of (i) *in-situ* charge detectors to monitor the voltages of bitlines and to estimate t_{RET} of bitcells, and (ii) an adaptive refresh and restore scheme to modulate the refresh rate and the restore timing parameters. The detailed circuit-level SPICE simulations confirm that our detector can detect the voltage of a bitcell in a 2-bit resolution. We evaluate the proposed architecture in an 8-core system using architectural simulators. The CDAR-DRAM can improve the system performance by 9.4% and energy efficiency of DRAM by 14.3%, as compared to the prior art [6].

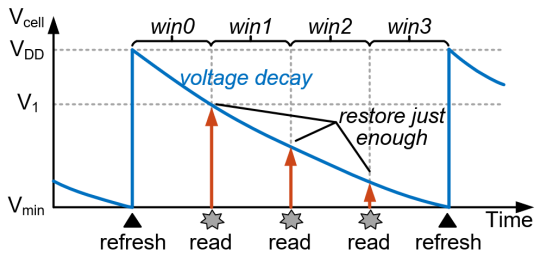

Fig. 1. The restore truncation scheme [6].

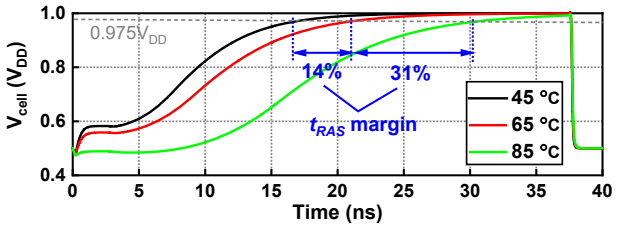

Fig. 2. The t_{RAS} margin for different temperatures.

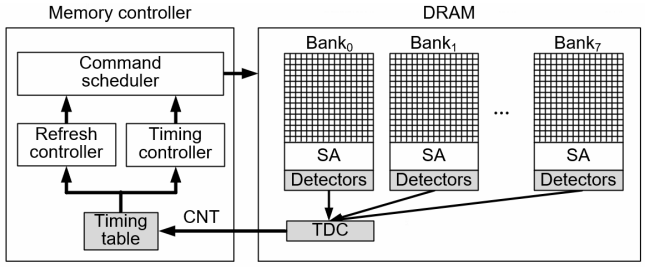

Fig. 3. The proposed CDAR-DRAM architecture.

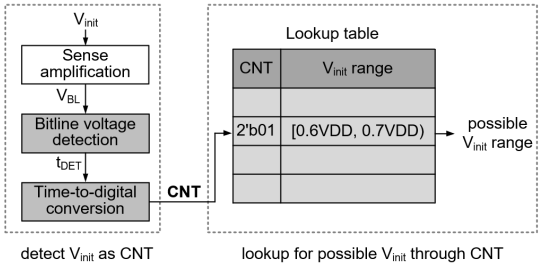

Fig. 4. The workflow of the *in-situ* charge detection scheme.

The major contributions of this paper can be summarized as follows: (1) We propose an *in-situ* bitline charge detection scheme that can estimate the DRAM cell's voltage in runtime. This helps to remove pessimistic timing margins for the cells that retain a good amount of charges. The detection process takes place simultaneously with DRAM's refresh operations, thereby incurring little performance overhead. (2) We propose an adaptive refresh and restore scheme. Working with the above detectors, the adaptive scheme can reduce refresh overhead and the pessimistic margin added to the restore timing. The scheme enables CDAR-DRAM to switch automatically between two modes: (i) a refresh mode to change the refresh interval progressively, and (ii) a restore mode to dynamically optimize restore timing. It provides improvements in both performance and energy efficiency.

The remainder of the paper is organized as follows. Sec. II presents the proposed CDAR-DRAM architecture. Sec. III describes our evaluation platform. Sec. IV introduces the experimental results and related discussion. Finally, Sec. V concludes the paper.

II. CDAR-DRAM ARCHITECTURE

In this section, we present CDAR-DRAM, an architecture with *in-situ* charge detection and adaptive refresh and restore schemes. Fig. 3 shows the proposed architecture. It requires modifying the DRAM and the controller. For a typical DRAM chip having multiple banks, each bank is composed of columns of bitcells and sense amplifier (SA) circuits. The proposed architecture adds bitline voltage detectors to bitlines, whose outputs are converted by a timing-to-digital converter (TDC). The TDC produces a 2-bit output CNT that roughly represents the smallest bitcell voltage of a row of bitcells. On the other hand, the controller takes CNT and estimates the maximum t_{REFW}, and updates a timing table based on it. After that, it issues commands with variable restore timing.

A. In-situ Charge Detection Scheme

Our goal is to find out the DRAM cell's voltage level right after the refresh window ($V_{c,init}$) so that we can reduce the refresh rate and shorten restore time if $V_{c,init}$ is high. To achieve this goal, we propose an *in-situ* charge detection scheme based on the detectors and the TDC.

Fig. 4 shows the workflow of our proposed *in-situ* charge detection scheme. First of all, the sense amplification is performed on $V_{c,init}$. If $V_{c,init}$ is $> 0.5 \cdot V_{DD}$, the sense amplification increases the bitline voltage (V_{BL}) toward V_{DD}. Here, the higher $V_{c,init}$ is, the faster the V_{BL} is restored to V_{DD}. Therefore, we measure the time when V_{BL} crosses a threshold voltage, t_{DET}, which can indicate the level of $V_{c,init}$ before the sense amplification. For being used in the memory controller, t_{DET} is converted to the digital value CNT, which the memory controller uses to estimate the $V_{c,init}$ level range based on a pre-defined lookup table.

Fig. 5(a) shows the circuits for the bitline voltage detection scheme. It illustrates the sensing of the capacitor voltage of the bitcell of the *i*-th column of a bank ($Cell_i$) via the bitline of the *i*-th column (BL_i). The existing SA circuits are recycled but we add the bitline detector. The detector contains three transistors: M0 is controlled by an "enable" signal, which is high for detection and low otherwise; M1 and M2 compose an inverter-based voltage comparator, whose input is BL_i and output is DET_i. The switching threshold voltage (V_M) of the comparator is set by the sizing of M1 and M2. We optimize the detector circuits for process variation tolerance, area, and power efficiency, which will be further discussed in Sec. II.B.

The detection takes place during the refresh operation. Specifically, a bitline (BL) is pre-charged to $0.5V_{DD}$ and then the access transistor of a bitcell is enabled. This initiates charge-sharing between a bitcell and a BL, which pushes V_{BL} by ΔV away from $0.5V_{DD}$. The SA amplifies this difference, and as a result, it *restores* the voltage of the bitcell capacitor and BL to V_{DD}. During this restoration process, the bitline detector compares V_{BL} with V_M and changes its output DET from high to low if V_{BL} is greater than V_M. The time that DET switches high to low is defined as t_{DET} and the TDC converts it to a digital code. Fig. 5(b) shows an example timing waveform of the bitline detection. We consider two bitcells $Cell_0$ and $Cell_1$, where we assume that $Cell_0$'s potential, denoted as $V_{c,init0}$ is high while $Cell_1$'s potential $V_{c,init1}$ is low. This difference makes BL_1 take longer in the restoration process than BL_0, making t_{DET1} longer than t_{DET0}.

Since the DRAM timing is constrained by the cell that has the lowest voltage in the row, we need to account for only the slowest DET_i switching. This can be achieved by OR-ing all the DET_i signals of a row (note that the DET_i signal is active low) and feeding its output DET_{slwst} to the TDC. Fig. 6 shows the exemplary waveform of the TDC process. Before the detection, CNT is zero initially. During the detection, CNT increases every clock cycle unless DET_{slwst} has fallen to zero. In the exemplary case, DET_1 transits later than DET_0 so the OR

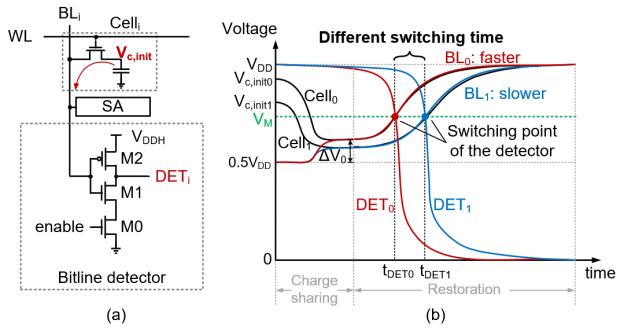

Fig. 5. (a) The hardware for bitline voltage detection; (b) The waveform of the bitline voltage detection for cells with high and low $V_{c,init}$.

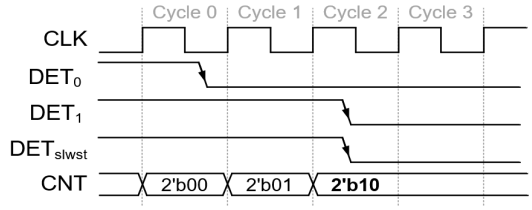

Fig. 6. The TDC process for converting the switching time of DET_{slwst} to CNT.

TABLE I. THE CORRESPONDING $V_{c,INIT}$ OF CNT

CNT	2'b11	2'b10	2'b01	2'b00
$V_{c,init}$ (V_{DD})	[0.55, 0.6)	[0.6, 0.7)	[0.7, 0.85)	[0.85, 1)

tree takes DET_1 as the DET_{slwst}. As a result, the final CNT is 2'b10 when DET_{slwst} transits low.

After obtaining the CNT, the corresponding V_{init} range can be inferred based on a pre-defined lookup table. We characterize the lookup table based on the circuit-level SPICE simulation. We found the CNT values for $V_{c,init}$ from the minimum sustainable voltage (V_{min}, set to $0.55V_{DD}$) to V_{DD}, which is summarized in Table I.

Note that so far the discussions assume DRAM cells store "1". The case for a cell storing "0" is similar except that we need to use the bitline detector whose V_M is less than $0.5V_{DD}$. Nevertheless, restoring a cell storing "0" is fundamentally faster than restoring a cell storing "1" due to the *NMOS* access transistor [19], thereby not as critical.

B. Design Optimization of The Bitline Charge Detector

To improve the robustness and energy efficiency, we optimize the bitline detector circuits in terms of transistor size and supply voltage V_{DDH}. First of all, the switching threshold V_M should be higher than $0.5V_{DD}$ for a cell storing "1". Upsizing M2 can increase V_M but it costs more silicon area. Increasing supply voltage (V_{DDH}) can also increase V_M yet at the cost of energy consumption and reliability.

Meanwhile, as shown in Fig. 7, the restore speed of the bitline slows down as V_{BL} approaches V_{DD}. This implies that large V_{M0} increases the variation of t_{DET}. For this respect, it is better to set V_M where the V_{BL} curve has the largest slope.

To find out the optimal parameters of the bitline detector, we perform circuit-level SPICE simulation. We build a 512×512 DRAM subarray along with SA circuits using 55nm DDR3 model parameters [20]. The transistor model is based on the Predictive Technology Model (PTM) [21]. To evaluate the robustness of the proposed detector, we run Monte Carlo simulations with process variations based on [22].

Fig. 8(a) shows the variation of t_{DET} (σ_t) across different transistor sizes and V_{DDH} values under a temperature of 85 ºC. The σ_t firstly

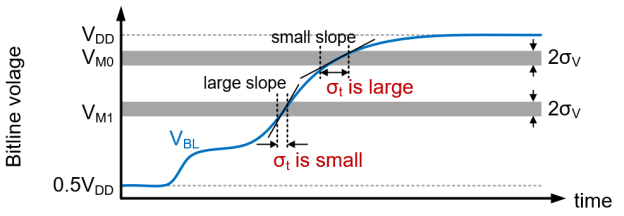

Fig. 7. The variation of the detector's output switching time σ_t is sensitive to the V_M of the detector.

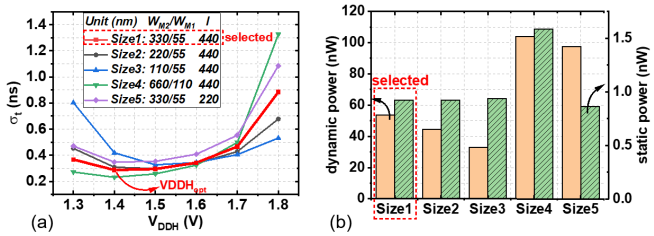

Fig. 8. (a) The variation of DET_i's switching time σ_t, and (b) the power consumption of the bitline detector.

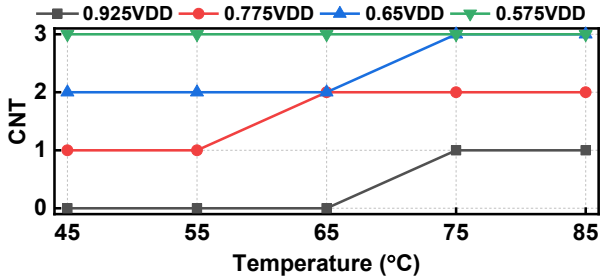

Fig. 9. The CNT value across different temperatures.

reduces and then increases due to the change of V_M. We set V_{DDH} to 1.4V since it achieves the smallest σ_t across most of the transistor sizes. Fig. 8(b) shows the dynamic and static power consumption with different transistor sizes. Considering reducing both variations and power consumption, the "Size1" is chosen for M1 and M2 of the detector. M0 is less important for the operation of the detector and it is set to the minimum size.

To show the robustness of the proposed detection circuits, we swept temperatures from 45 ºC to 85 ºC and check the variation of CNT. Fig. 9 shows the simulated CNT with different $V_{c,init}$s. For $V_{c,init}$=0.575VDD, the detection circuits generate the CNT correctly (CNT=2'b11) for all temperatures, therefore ensuring the correct operations for the worst-case cells. When $V_{c,init}$ is higher, the detection circuits generate CNT correctly when the temperature is 45 ºC to 55 ºC. As the temperature goes up, the CNT may be one level higher than expected, which results in a pessimistic estimation of $V_{c,init}$.

C. Adaptive Refresh and Restore Scheme

So far, we obtained the CNT as the representative of the possible range of $V_{c,init}$, with which the memory controller can remove the existing timing margins for cells with high $V_{c,init}$.

To improve DRAM performance and energy efficiency, we proposed a multi-rate refresh and adaptive restoration scheme, which contains (i) a data retention time estimation method to find out the proper refresh rates and (ii) a timing-table-based adaptive refresh and restore strategy.

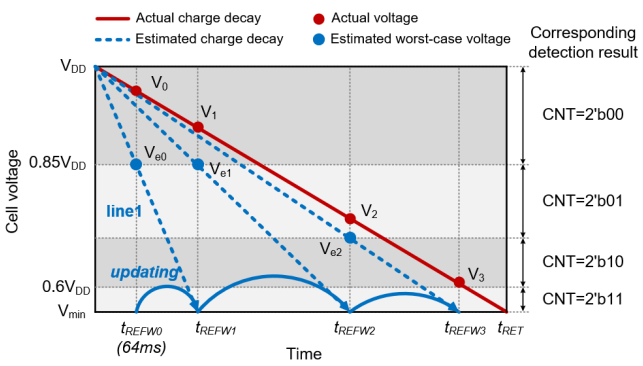

Fig. 10. The retention time estimation process.

Fig. 10 shows an example of our presented retention time estimation process, which progressively increases the refresh window (t_{REFW}) until it approaches the data retention time (t_{RET}) of a DRAM cell for minimizing the refresh rate. The red line shows the actual DRAM cell voltage would gradually decay from V_{DD} to the minimum sustainable voltage V_{min} within t_{RET}, which can be approximated as a linear curve [6]. The memory controller issues a refresh operation every time at t_{REFWX} (X is the iterations of our estimation process, which can be 0, 1, 2, or 3), where the cell voltage decays from V_{DD} to V_X (i.e., $V_{c,init}$ at Xth iteration).

Initially, refresh operations are sent at t_{REFW0} of 64ms. The detection circuits would output a CNT of 2'b00 after the first detection at t_{REFW0}, indicating V_0 is between $0.85V_{DD}$ and V_{DD} (see Table I). The low bound of possible V_0 (i.e., V_{e0}=$0.85V_{DD}$) is used for the worst-case charge decay estimation as shown with the "line1". The horizontal intercept of "line1" is taken as t_{REFW1}, where the cell voltage is estimated to decrease to V_{min}. Then, t_{REFW1} is set as the new refresh window for the next iteration.

Repeating the process until the CNT at t_{REFWX} is 2'b10. As a result, V_X is low to $0.6V_{DD}$, which is very close to V_{min} (=$0.55V_{DD}$), indicating that the final t_{REFWX} is nearly the maximum. Due to temperature variations, the CNT may become 2'b11, indicating a very low V_X. The V_X may be lower than V_{min} if the temperature goes up, so the t_{REFW} is reduced by half for safety once CNT is 2'b11.

During the above estimation process, the t_{REFW} of each step is taken from the horizontal intercept of the approximated linear charge decay, which is formalized as follows:

$$t_{REFW} = t_{REFWpre} \times (V_{DD}-V_{min})/(V_{DD}-V_{c,init,lb}) \quad (1)$$

where $t_{REFWpre}$ is the previous refresh window time, and $V_{c,init,lb}$ is the lower bound of possible $V_{c,init}$ according to CNT.

By the introduced retention time estimation method, the maximum t_{REFW} can be approached during runtime without stalling DRAM. The proposed method can adaptively change t_{REFW} against temperature variations to achieve a sufficient and reliable refresh rate reduction.

Based on the estimated t_{REFW}, a timing-table-based dynamic refresh strategy is shown in Fig. 11 (❶, ❶, and ❷). The timing table (in the refresh mode) uses a 3-bit tag ("001"/"01X"/"1XX") to represent the refresh rate for each row. The position of the first "1" in the tag represents a different t_{REFW}. The right-side bits (e.g., "XX") are used as a refresh counter, which decreases every 64ms (❶). If the refresh counter is zero, the memory controller sends a refresh operation. The tag is initialized to "001" (❶) for a t_{REFW} of 64ms and will be updated due to suboptimal $t_{REFWpre}$ (❷). For example, if the estimated t_{REFW} can be twice of $t_{REFWpre}$, a tag of "01X" (128ms- row) is updated to "1XX" (256ms-row) to reduce the refresh rate.

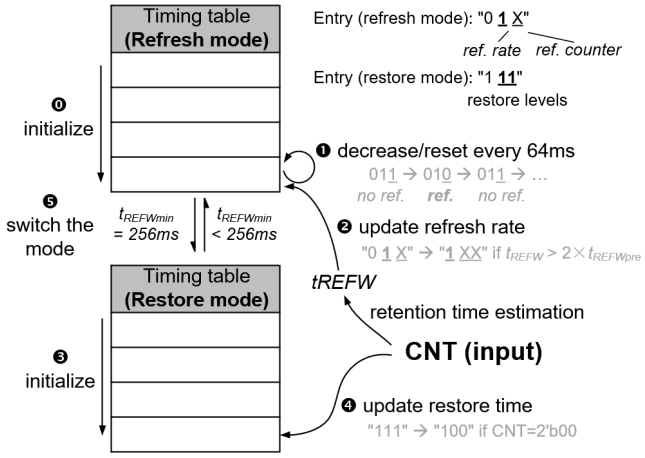

Fig. 11. A timing-table-based adaptive refresh and restore strategy.

TABLE II. ADJUSTED REFRESH RATE AND RESTORE TIME

Mode flag	Tag	t_{REFW} (ms)	t_{RAS} for 1st/2nd/3rd/4th sub-window (cycles)	t_{WR} for 1st/2nd/3rd/4th sub-window (cycles)
Refresh mode ("0")	001	64	32/24/20/19	14/12/10/9
	01X	128		
	1XX	256		
Restore mode ("1")	111	256	32/24/20/19	14/12/10/9
	110	256	28/20/16/16	
	101	256	23/16/13/13	
	100	256	16/13/13/13	

Since most cells have high $V_{c,init}$s at low temperatures, an adaptive restore scheme (❸ and ❹ in Fig. 11) is adopted to speed up restorations for these cells. In restore mode, the low two bits of the tag represent different restore speed levels. All tags are initialized to "111" (❸) for a conservative restore time. The tag will be updated for optimal restore time (❹) according to CNT. For example, if CNT is 2'b00, which means the cell has a high $V_{c,init}$ (≥$0.85V_{DD}$), a tag of "111" will be updated to "100" for faster restoration of the cell.

The CDAR-DRAM can automatically switch between the refresh mode and restore mode (❺). The memory controller starts with the refresh mode and tracks the minimum t_{REFW} ($t_{REFWmin}$) in the timing table. If $t_{REFWmin}$ is 256ms (i.e., all tags are "1XX"), the memory controller will switch automatically to the restore mode and setting all tags to "111" (❸). Moreover, if the t_{REFW} has to be reduced (i.e., CNT=2'b11), the CDAR-DRAM switches back to the refresh mode.

Table II lists the adjusted restore timing parameters and the refresh rate for different tags through a circuit-level SPICE simulation. Besides, similar to the restore truncation scheme [6], for each tag, four different t_{RAS}/t_{WR} are adopted for operations that arrive in the four sub-windows of t_{REFW} (see Fig. 1). Note that the minimum t_{RAS} in the table is 13 cycles since t_{RAS} should be larger than t_{RCD} (12 cycles). The t_{RAS} could be even lower if reducing t_{RCD} through methods in [3-5, 16-18] to further speed up the restoration.

III. EVALUATION METHODOLOGY

To evaluate the proposed CDAR-DRAM architecture, we run the simulation using a cycle-accurate DRAM simulator, Ramulator [23]. For workloads, we randomly selected benchmarks from SPEC CPU2006 and PARSEC suite, covering non-memory-intensive ones

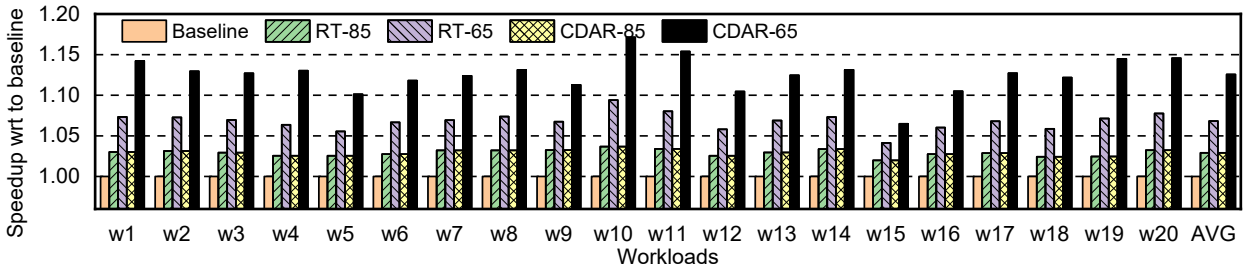

Fig. 12. Performance comparison of different schemes.

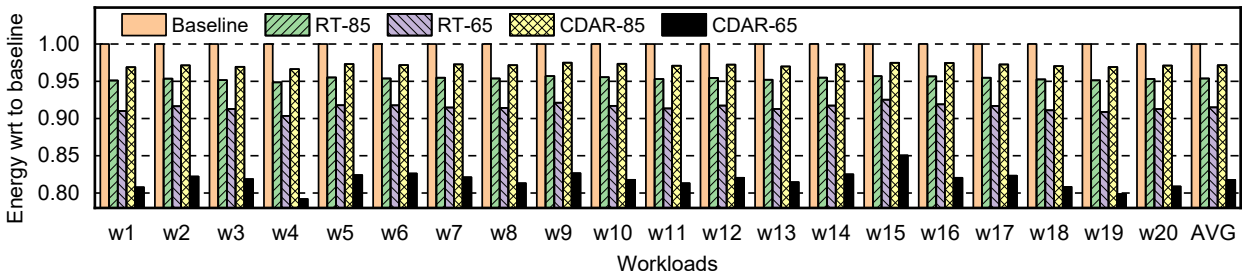

Fig. 13. Energy comparison of different schemes.

like *444.namd*, *458.sjeng*, and *blackscholes*, as well as memory-intensive ones like *429.mcf*, *470.lbm*, and *canneal*. The benchmarks are converted to CPU traces for Ramulator through Pin [24]. We use DRAMPower [25] to obtain the energy results of DRAM.

We modeled an 8-core system with configurations shown in Table III. The DRAM timing parameters follow the Micron DDR3 datasheet [2]. We used the DRAM retention distribution parameters provided in [12] and modeled the temperature effects according to [13]. The simulation was performed for 1 billion cycles, corresponding to 256ms given our 4GHz clock frequency, which ensures each row is refreshed at least once considering a maximum t_{REFW} of 256ms. We measured the performance using the weighted speedup metric [27]. The energy results were reported as the average energy consumption per request [1].

TABLE III. SYSTEM CONFIGURATION

Processor	8-core, 4GHz, 3-wide issue, 128 ROB size
Memory Controller	Address mapping: rw:cl:rk:bk:ch:offset 32-entry read/write request queues Page management policy: closed-page with FR-FCFS
DRAM	DDR3-1866 [2] 2 channels, 1 rank/channel, 8 banks/rank 64K rows/bank, 8 KB/row tCK = 1.07ns, width: x8 $t_{RCD}/t_{RAS}/t_{WR}$ = 12/32/14 cycles

IV. RESULTS AND ANALYSIS

To evaluate the effectiveness of CDAR-DRAM, we made a comparison with the normal DDR3 [2] and the RT architecture [6]. The RT architecture adopted both multi-rate refresh and partial restoration methods to optimize the refresh window (t_{REFW}) and restore time (t_{RAS}), which is similar to ours. There are also plenty of prior works that tried to optimize the refresh or restoration operations, however, they focus on improving only the refresh [1, 8-11,15] or restoration [3-5, 16-18]. Besides, some prior works [4, 19] further optimize restoration beyond the RT architecture by adopting the t_{RCD} reduction methods, which are orthogonal with ours. For the above reasons, we chose to make a comparison with the state-of-the-art RT architecture [6].

We evaluated the normal DDR3, RT architecture, and the proposed CDAR-DRAM under two different conditions: the worst-case condition (the temperature is 85 ºC) and the normal-case condition (the temperature varies between 45 ºC and 65 ºC [16, 28]). For the normal-case condition, we assumed the temperature is 45 ºC for 50% of the time and 65 ºC for the remaining, which is a pessimistic assumption for our design since the DRAM temperature may seldom exceed 50 ºC [16].

The following architectures were studied:

- Baseline: the normal DDR3 [2] with timing parameters that ensure the data correctness in the worst-case temperature, i.e., 85 ºC.
- RT-85/-65: the RT DRAM architecture with fixed timing parameters that are profiled in the worst-case/normal-case condition.
- CDAR-85/-65: the proposed CDAR-DRAM at the worst-case/normal-case temperature.

A. Profiling Overhead

To mitigating refresh or restore overhead, most prior works [1, 6, 8-11, 14, 15] have to find the data retention time t_{RET} of all cells through profiling. The profiling process is very time-consuming and the DRAM has to be laid off during profiling. According to the experiment for 2Gb DRAM chips in [13], the profiling process takes *days* to reliably find the lowest t_{RET} of all cells. Due to the error accumulation [14] and long-term issues like aging, the profiling process has to perform termly. However, in the proposed CDAR-DRAM, the t_{RET} can be estimated through *in-situ* detection simultaneously with the refresh operations, which takes no extra time.

B. Impact on Performance

Fig. 12 shows the normalized weighted speedup on 20 randomly combined eight-core workloads (i.e., from w1 to w20) and the average result (i.e., AVG). When the temperature is 85 ºC, where DRAM cells have small t_{RET}s, the CDAR-85 works in the refresh mode, whose timing table represents different refresh rates as RT-85 does. So, CDAR-85 and RT-85, on average, have the same 2.9% performance improvement over the baseline.

In the normal condition, the normal DDR3 still has to use the worst-case timing parameters. For the RT-65, we assume that it can be aware of the DRAM temperature and is profiled at 65 ºC for the normal condition. By removing the timing margin of t_{RAS} from 85 ºC to 65 ºC, RT-65 achieves an average 6.4% performance improvement over the baseline.

Since our proposed CDAR-65 can modulate timing parameters adaptively through the *in-situ* detection circuits with temperature variations in the normal condition, the system performance is improved by 12.6% over the baseline. The CDAR-65 showed a performance improvement of 9.4% and 5.4% over the RT-85 and RT-65, respectively. The CDAR-65 outperformed the RT architecture because of the unexploited timing margins by the static profiling method. If the DRAM temperature is 45 ºC or even below [16], most DRAM cells have t_{RET} greater than 780ms [13] therefore can retain more charge on their capacitor. In this case, a more aggressive reduction of *tRAS* could make for restoring these cells in our CDAR architecture.

C. Impact on DRAM Energy

Fig. 13 compares the energy consumption of different architectures on the 20 workloads. At 85 ºC, the RT-85 showed a 4.6% energy reduction as compared to the baseline. The CDAR-85 reduced energy by 2.8% than baseline. The CDAR-85 consumed more energy than the RT-85 since the detection circuits of CDAR-85 would cause additional energy overhead. In the normal-case condition, the RT-65 showed an energy reduction of 8.5%. The proposed CDAR-65 can save energy by 18.2% compared to the baseline. As compared with RT-85/-65, CDAR-65 can reduce energy consumption by 14.3%/10.6%.

D. Area Overhead of Detection Circuits

We evaluated the area overhead of our detection circuits in a DRAM chip since the DRAM is very cost-sensitive. The detection circuits contain two parts: bitline detectors and the TDC block. Based on the size of detectors (see Fig. 8) and the transistor sizes of SA circuits from [20], the area of bitline detectors is about 9.13% of the area of the SA circuits. Considering the sense amplifier circuits occupy 8%~15% of the DRAM die area [26], our detectors cause up to 1.37% area overhead. Besides, according to the synthesis results of our TDC block with a 65nm process, the TDC block takes an extra 0.38% area overhead of a DRAM die. Thus, the proposed detection circuits increase the DRAM die area by 1.11%~1.75%.

V. Conclusion

In this paper, we proposed CDAR-DRAM, an architecture that adaptively adjusts refresh rate and relaxes restore timing based on *in-situ* detecting of the bitline voltage. Unlike prior profiling methods to collect data retention time under the static worst-case condition, the CDAR-DRAM adopted *in-situ* skewed-inverter-based detection circuits to estimate the retention time of cells during runtime without stalling normal DRAM operations. Based on the proposed adaptive refresh and restore scheme, the CDAR-DRAM can dynamically adjust the refresh rate and restore time for different cells. The proposed CDAR-DRAM can reduce the pessimistic timing margins that are preserved for the worst-case temperature in a rare case and a small number of weak cells. Experimental results show that in a typical temperature range, CDAR-DRAM can improve the system performance by 9.4% and saves energy by 14.3% for an 8-core system as compared with the state-of-art architecture.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 61774104), the National Key Research and Development Program of China (No. 2018YFB2202004), and US National Science Foundation (1453142 and 1919147).

Reference

[1] J. Liu, B. Jaiyen, R. Veras, and O. Mutlu, "RAIDR: Retention-aware intelligent DRAM refresh," in *ISCA*, Jun. 2012, pp. 1-12.

[2] Micron Technology, "4Gb: x4, x8, x16 DDR3 SDRAM," 2011.

[3] D. Lee *et al.*, "Tiered-latency DRAM: A low latency and low cost DRAM architecture," in *HPCA*, 2013, pp. 615-626.

[4] Y. Wang *et al.*, "Reducing DRAM Latency via Charge-Level-Aware Look-Ahead Partial Restoration," in *MICRO*, 2018, pp. 298-311.

[5] H. Hassan *et al.*, "ChargeCache: Reducing DRAM latency by exploiting row access locality," in *HPCA*, 2016, pp. 581-593.

[6] X. Zhang, Y. Zhang, B. R. Childers, and J. Yang, "Restore truncation for performance improvement in future DRAM systems," in *HPCA*, 2016, pp. 543-554.

[7] U. Kang *et al.*, "Co-architecting controllers and DRAM to enhance DRAM process scaling," in *The memory forum*, 2014.

[8] I. Bhati, Z. Chishti, S. Lu, and B. Jacob, "Flexible auto-refresh: Enabling scalable and energy-efficient DRAM refresh reductions," *in ISCA*, 2015, pp. 235-246.

[9] A. Agrawal, A. Ansari, and J. Torrellas, "Mosaic: Exploiting the spatial locality of process variation to reduce refresh energy in on-chip eDRAM modules," in *HPCA*, 2014, pp. 84-95.

[10] S. Liu, K. Pattabiraman, T. Moscibroda, and B. G. Zorn, "Flikker: Saving DRAM refresh-power through critical data partitioning," in *Proc. ASPLOS*, 2011.

[11] R. K. Venkatesan, S. Herr, and E. Rotenberg, "Retention-aware placement in DRAM (RAPID): software methods for quasi-non-volatile DRAM," in *HPCA*, 2006, pp. 155-165.

[12] K. Kim and J. Lee, "A New Investigation of Data Retention Time in Truly Nanoscaled DRAMs," *IEEE Electron Device Letters*, vol. 30, no. 8, pp. 846-848, Aug. 2009.

[13] J. Liu, B. Jaiyen, Y. Kim, C. Wilkerson, and O. Mutlu, "An experimental study of data retention behavior in modern DRAM devices: Implications for retention time profiling mechanisms," *ACM SIGARCH Computer Architecture News*, 41(3), 60-71. 2013.

[14] M. Patel, J. S. Kim, and O. Mutlu "The Reach Profiler (REAPER) Enabling the Mitigation of DRAM Retention Failures via Profiling at Aggressive Conditions," *ACM SIGARCH Computer Architecture News*, 45(2), 255-268. 2017.

[15] A. Das, H. Hassan, and O. Mutlu "VRL-DRAM: Improving DRAM Performance via Variable Refresh Latency," in *DAC*, 2018, pp. 1-6.

[16] D. Lee *et al.*, "Adaptive-latency DRAM: Optimizing DRAM timing for the common-case," in *HPCA*, 2015, pp. 489-501.

[17] W. Shin, J. Yang, J. Choi, and L. Kim, "NUAT: A non-uniform access time memory controller," in *HPCA*, 2014, pp. 464-475.

[18] K. Chandrasekar *et al.*, "Exploiting expendable process-margins in DRAMs for run-time performance optimization," in *DATE*, 2014, pp. 1-6.

[19] H. Luo *et al.*, "CLR-DRAM: a low-cost DRAM architecture enabling dynamic capacity-latency trade-off," in *ISCA*, 2020, pp. 666-679.

[20] Rambus, "DRAM power model (2010)," http://www.rambus.com/energy.

[21] PTM, "Predictive technology model," http://ptm.asu.edu.

[22] K. Chandrasekar, C. Weis, B. Akesson, N. Wehn, and K. Goossens, "Towards variation-aware system-level power estimation of DRAMs: An empirical approach," in *DAC*, 2013, pp. 1-8.

[23] Y. Kim, W. Yang, and O. Mutlu, "Ramulator: A Fast and Extensible DRAM Simulator," in *IEEE Computer Architecture Letters*, vol. 15, no. 1, pp. 45-49, 1 Jan.-June 2016.

[24] C. K. Luk, *et al.*, "Pin: building customized program analysis tools with dynamic instrumentation, " *ACM sigplan notices*, 40(6), 190-200. 2005.

[25] K. Chandrasekar *et al.*, "DRAMPower: Open-source DRAM Power & Energy Estimation Tool," http://www.es.ele.tue.nl/drampower/, 2012.

[26] T. Vogelsang, "Understanding the energy consumption of dynamic random access memories," in *MICRO*, 2010, pp. 363-374.

[27] A. Snavely and D. M. Tullsen, "Symbiotic job scheduling for a simultaneous multithreading processor," in ASPLOS-IX, 2000.

[28] N. El-Sayed, *et al.*, "Temperature management in data centers: Why some (might) like it hot", in *Technical Report*, *CSRG-615*, 2012.