Improving Energy Efficiency of DRAM by Exploiting Half Page Row Access

Heonjae Ha*, Ardavan Pedram*†, Stephen Richardson*, Shahar Kvatinsky‡, and Mark Horowitz*
*Stanford University †Movidius ‡Technion – Israel Institute of Technology
{hunjaeha, perdavan, steveri, horowitz}@stanford.edu shahar@ee.technion.ac.il

***Abstract*—DRAM energy is an important component to optimize in modern computing systems. One outstanding source of DRAM energy is the energy to fetch data stored on cells to the row buffer, which occurs during two DRAM operations, *row activate* and *refresh*. This work exploits previously proposed half page row access, modifying the wordline connections within a bank to halve the number of cells fetched to the row buffer, to save energy in both cases. To accomplish this, we first change the data wire connections in the sub-array to reduce the cost of row buffer overfetch in multi-core systems which yields a 12% energy savings on average and a slight performance improvement in quad-core systems. We also propose charge recycling refresh, which reuses charge left over from a prior half page refresh to refresh another half page. Our charge recycling scheme is capable of reducing both auto- and self-refresh energy, saving more than 15% of refresh energy at 85°C, and provides even shorter refresh cycle time. Finally, we propose a refresh scheduling scheme that can dynamically adjust the number of charge recycled half pages, which can save up to 30% of refresh energy at 85°C.**

I. Introduction

About a decade ago voltage scaling slowed, making energy the primary limitation for both servers and mobile computing systems. A major source of energy consumption in such computing systems is DRAM, which consumes more than 25% of the system energy in data centers [1]. This paper focuses on the DRAM energy used to fetch data to the row buffer, which is the most energy inefficient operation of DRAM in modern multi-core systems.

A DRAM access starts by fetching data stored on thousands of cells and sending it to the row buffer. Each subsequent memory request then transfers less than 1% of data held in the row buffer to the processor or vice versa. In modern multi-core systems, less than 4% of the data pre-loaded to the row buffer are accessed before closing it [2]. This is due to the interleaved memory requests between different applications, which severely degrades the row buffer hit rate. Hence, most of the energy used to fetch data to the row buffer is wasted. This phenomenon is often called *row buffer overfetch* [3], [4].

Refresh is the other DRAM operation that utilizes the row buffer. A DRAM stores charge on a cell capacitor to represent the data. Those charges leak over time and must be refreshed periodically. Because there are billions of cells in modern Gb-scale DRAM, the row buffer is used to refresh thousands of cells simultaneously. The energy consumed to refresh cells is becoming increasingly important as mobile computing systems become more popular; mobile devices are often placed in low power sleep mode for long periods of time where the energy of refresh dominates the total system energy [5].

978-1-5090-3508-3/16/$31.00 © 2016 IEEE

Recent DDR4 supports ×4 half page architecture [6], which reduces the amount of data fetched to the row buffer to half that of DDR3. Using this DDR4 feature as a baseline, we add simple modifications to provide a unique sub-array structure that reduces both row buffer overfetch and refresh energy cost.

This paper makes the following contributions:

- We investigate cost-per-bit optimizations in modern DRAMs, i.e. (i) hierarchical wordline and data wires and (ii) staggered wordline drivers and sense amplifiers, and show how they constrain potential DRAM modifications.
- We extend the current DDR4's half page architecture to achieve full DRAM bandwidth for I/O organizations other than ×4, and show that accessing half page rows reduces row buffer overfetch cost, which improves energy efficiency of DRAM in multi-core systems.
- We propose charge recycling refresh that reduces both auto- and self-refresh energy by recycling charges from a prior half page refresh to refresh another half page. We then explore a refresh scheduling scheme that recycles charges continuously across multiple half pages to save even more energy. The effect and practical use of multiple row charge recycling refresh are also discussed.

To better understand the constraints posed by today's DRAM, Section II reviews the structure and operation of modern commodity DRAM. With this background, we propose a new sub-array structure that allows half page row access along with two extensions that improve energy efficiency of DRAM in Section III, followed by detailed analysis in Section IV. Section V presents our evaluation methodology, and the proposed solution's effectiveness is shown in Section VI. How this approach improves on prior work is presented in Section VII.

II. Background

Modern commodity DRAM has a hierarchical organization where an array of memory cells form a *MAT* and a collection of MATs are grouped into a *sub-array* [7]. Sub-arrays are then stacked on top of each other to form a *sub-bank* and, depending on the size of the DRAM, one or more sub-banks combine to form a *bank*. Finally, there are multiple banks on a DRAM chip and the memory module that we use in computer systems has multiple DRAM chips to provide 64 bits of I/O.

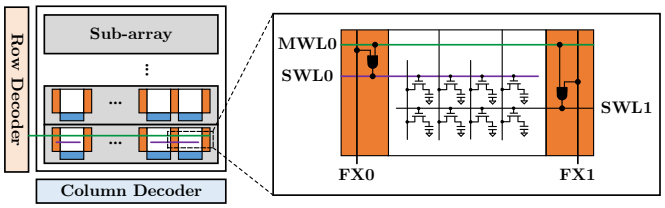


Fig. 1. Hierarchical wordline structure within a bank. Main wordline (MWL) and pre-decoded row address wires (FX) select sub-wordlines (SWL). Sub-wordline drivers are located in the orange rectangles.

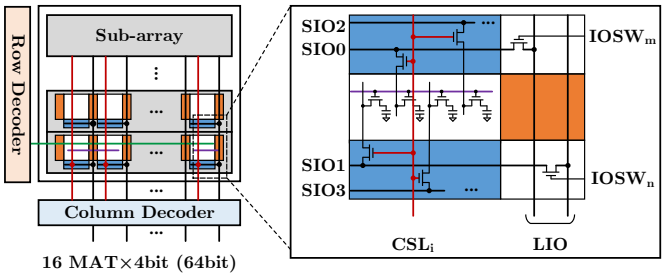


Fig. 2. Hierarchical data wire connection within a bank. Data transfers from cell to column decoder in the order of bitline, segmented I/O (SIO), and finally local I/O (LIO) wires. Sense amplifiers are in the blue rectangles.

Data stored on DRAM is accessed using a sequence of three commands, namely activate, read/write, and precharge. *Activate* selects a wordline along with the page-size group of cells connected to it. Data stored on selected cells are sensed and amplified by the sense amplifiers, also known as the row buffer. *Read/write* then selects a small portion of the data held in the row buffer and either transfers it to the periphery or updates its value. Once data stored on cells of the selected wordline have been fully restored by the row buffer, *Precharge* is issued to deselect the wordline and isolate cells from the bitlines, which allows one to activate a different wordline.

A. Wordline Selection

The activate command selects a wordline that spans typically the whole length of the sub-array. If a single wire was used to drive such a large load, the RC delay of the wire would significantly slow wordline transitions. To avoid this delay, a wordline is divided into multiple shorter segments called sub-wordlines (SWL) organized in a hierarchical wordline structure [8] which reduces the RC loading of each wire.

Sub-wordline drivers are located on both sides of the MAT and each SWL is selected with a main wordline (MWL) and a pre-decoded row address (FX) [6], as shown in Figure 1. Because the pitch of a SWL is tight, sub-wordline drivers are placed in a staggered or interleaved manner for optimal layout area [9]. Also to further reduce chip area, sub-wordline drivers except those at the edges drive both left and right MATs [6].

B. Data Transfer within a Bank

Data wires are shared among many different cells in a bank by forming a hierarchical structure of bitline, segmented I/O (SIO), and local I/O (LIO) [10], as shown in Figure 2. SIO runs over the sense amplifiers, and routes the output of selected bitlines to the LIO. LIO runs vertically and connects

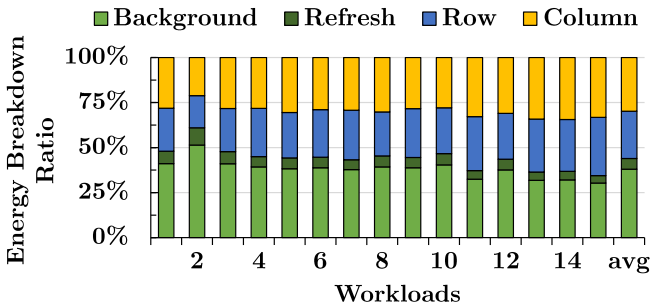


Fig. 3. DRAM energy breakdown of selected workloads. System configuration and workload details are described in Section V.

the SIO of the selected sub-array to the periphery through I/O switch (IOSW) transistors. The gates of these switch transistors are driven by a signal from the row decoder. Similar to the wordline driver, sense amplifiers are also staggered or interleaved [9], [10] where half of the bitlines connect to sense amplifiers placed above the cell arrays and the other half of the bitlines connect to sense amplifiers below the cell arrays. As a result, sense amplifiers are shared between two sub-arrays and two sets of IOSW wires (IOSW$_m$ and IOSW$_n$ in Figure 2) are selected to properly connect all even and odd SIO wires to LIO wires.

A read operation transfers data out of the bank by connecting a small number of bitlines to SIO and LIO wires using the column select lines (CSL). CSL is decoded from the column address and each MAT has one CSL selected that connects 4 bitlines to SIO. Hence, full I/O bandwidth for $\times 8$ I/O organization DDR4 is only achieved when all 16 MATs within a sub-array transfer data out of the bank [11]. This is because DDR4 has burst size of 8, so a total of 64 bits of data has to be transferred with $\times 8$ I/O organization. A write operation transfers data in to the bank by first driving the LIO and SIO wires with updated values. Then CSL shorts SIO wires to the corresponding bitlines, which will update the cell that it is connected to.

C. DRAM Energy Breakdown

In this work, we classify DRAM energy into four categories: background, refresh, row, and column. Figure 3 shows this energy breakdown, collected by simulating workloads in quad-core systems as described in Section V.

Background energy accounts for 38% of DRAM energy, as shown in Figure 3. It includes both static and dynamic standby energy, which depend on how long the banks were idle or activated and how long the ranks were placed in power down mode.

Refresh energy is the energy dissipated to perform periodic refresh. Temperature is an important factor in determining refresh energy because it affects the frequency of refreshes. Refresh energy is on average 6% of the total at 85°C for the workloads simulated.

Row energy includes energy dissipated to perform activate and precharge operations, which is 26% on average. Charging and discharging the bitlines along with the sense amplifiers

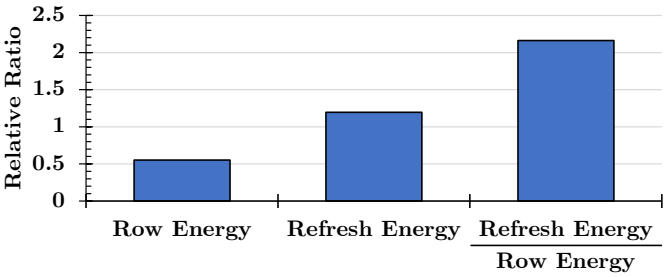


Fig. 4. Ratio of row, refresh, and refresh/row energy for 4Gb DDR4 relative to 4Gb DDR3 that are manufactured from the same company.

are one of the major source of row energy. This is because thousands of bitlines and sense amplifiers are accessed during each row cycle.

Column energy is the energy dissipated to transfer data between the row buffer and DRAM interface. Data bus termination and the read driver energy are also included, which depend on the termination configuration. For the workloads simulated, column energy averages 30% of the total.

D. DRAM Refresh

A DRAM is periodically refreshed to retain data stored on cells. The requirement for refresh is provided by standards and is usually represented as the number of refresh commands that have to be issued within a certain time window (tREFW). These two parameters and the number of rows determine the interval between refresh commands (tREFI) and the refresh cycle time (tRFC). As DRAM density increases, more rows are usually added to maintain the row buffer page size. This means that more rows are refreshed during each refresh command and the time to perform that refresh or tRFC increases due to power constraints.

Modern DRAM uses auto-refresh, where the DRAM determines the row to be refreshed and the memory controller just needs to issue refresh commands to indicate when the refresh should occur. If some rows have reliability issues due to noise, auto-refresh lets the DRAM refresh those rows more often, and so avoid problems like row hammer [12].

Another type of refresh is self-refresh, which refreshes data when the DRAM is in a low power sleep mode. In this mode, refreshes are automatically issued by the DRAM without any external command from the memory controller. And as self-refresh mode continues longer, static standby energy is suppressed considerably because temperature also decreases with DRAM in low power sleep mode. As a result, refresh energy is a significant portion of all energy used while in self-refresh [5].

Refresh energy is becoming even more important as technology nodes continue to shrink. Figure 4 compares dynamic row and refresh energy for a 4Gb DDR4 and DDR3 manufactured by the same company. Even though row energy of DDR4 decreased by half compared to DDR3, refresh energy is slightly larger. The refresh specifications and the number of rows refreshed in each refresh cycle are the same for both DDR4 and DDR3 according to the standards [13], [14]. Hence,

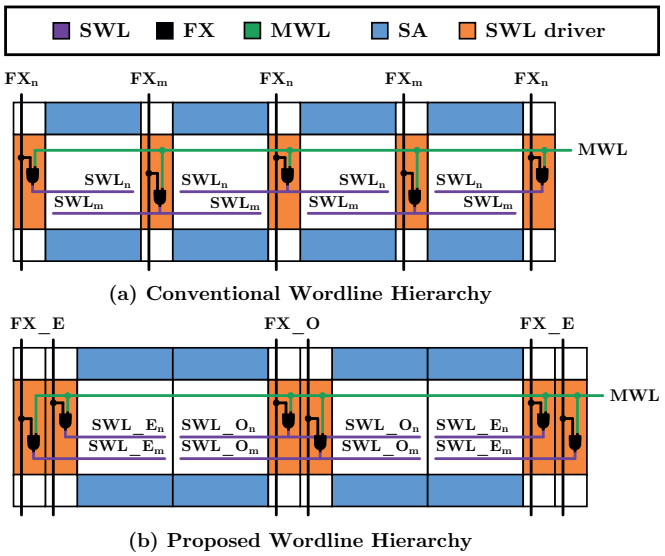


Fig. 5. Comparing conventional sub-array and proposed sub-array design that enables half page access by modifying the wordline hierarchy: (a) Conventional wordline hierarchy and (b) proposed wordline hierarchy where every sub-wordline of a MAT is driven from one side of the MAT.

we conjecture that twice as many cells are being refreshed in each refresh cycle for newer technology node devices to compensate for the yield loss. We also believe that this trend will continue, making refresh energy one of the most important uses of energy in DRAM.

III. Proposed Design

This section begins by describing a new sub-array structure that can selectively access half of a page. On top of this new sub-array we then build two new schemes, *Half Page DRAM* and *Charge Recycling Refresh*, to reduce the row buffer overfetch cost and refresh energy. We also discuss changes in interface protocols and refresh scheduling methods to fully utilize the potentials of the proposed schemes.

A. Reorganizing the Sub-array

We propose a new sub-array structure that segments the wordline in half using the $\times 4$ half page architecture of DDR4 [6] as a baseline. Our sub-array is fully compatible with a conventional DRAM and users can switch from half to full page by setting a DRAM register using MRS commands.

Like the baseline, our sub-array will double the pre-decoded row address (FX) wires, bitline equalization wires, and enable signals for sense amplifier supply voltages over the conventional sub-array shown in Figure 5a. In DDR4 these extra signals are used to segment the sub-array into a right and left half, where the right 8 MATs use one set of signals and the left use the alternate set. This partition of the sub-array makes it possible to access half the sub-wordlines in a row.

We more finely interleave these same resources as shown in Figure 5b. To accomplish this we need to move the wordline drivers that were previously on both sides of the MAT to the same side, and alternate odd and even FX signals to these columns of local wordline drivers. Notice that both the

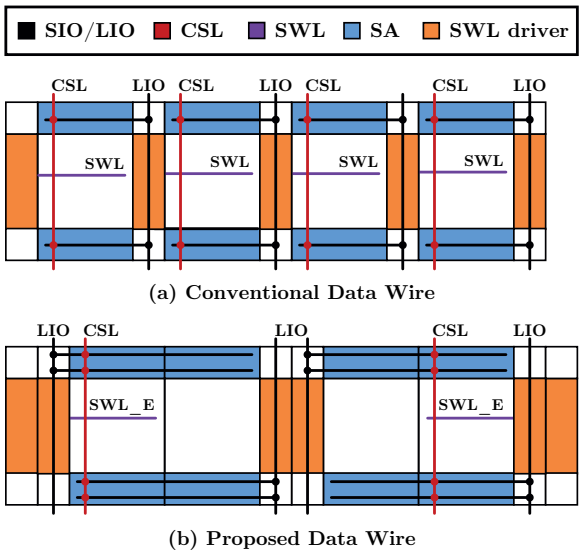


Fig. 6. Comparing conventional sub-array and proposed sub-array of Half Page DRAM that is built on top of the new sub-array shown in Figure 5b: (a) conventional data wire connections and (b) proposed data wire connections where number of SIO is doubled.

number and pitch of the wordline drivers are unchanged in this transformation, which avoids any area overhead. Like the baseline, activating either the odd or even FX lines enables one to access half of the sub-worldlines in a row. However, unlike the FX connections of the baseline, the FX selecting half of the sub-wordlines (FX_E) and the other half (FX_O) are connected to alternating pairs of wordline drivers. In our design, one of the MATs neighboring the selected MAT is always unselected, which is unique to this structure, and key to the charge recycling refresh described in Section III-D.

B. Half Page DRAM

The new sub-array structure described in the previous section cuts the page size of $\times 4$ I/O in half so that the same page size row buffer is used regardless of I/O organization. In other words, it does not reduce the row buffer overfetch cost, but removes the inherent inefficiency of DDR3's $\times 4$ I/O. When the new sub-array structure is used for $\times 8$ and $\times 16$ I/O organization, bandwidth reduces significantly because only half the LIO wires are active, so half as many bits are transferred from the sub-array with each access [15]. To reduce row buffer overfetch cost while maintaining full bandwidth, we create *Half Page DRAM* by making a small change to the I/O wiring in our previous sub-array.

Transferring more data from each MAT can be done by either doubling SIO wires or LIO wires. We choose to double SIO wires as shown in Figures 6a and 6b to minimize area overhead. Because SIO wires are doubled, only half as many bitlines within a MAT connect to each SIO wire, so we are able to double the length of these wires, enabling them to span 2 MATs. While this increases the wire capacitance, the number of bitline connections remains constant so the capacitance only increases slightly. SIO to LIO wire connections are also modified so that 4 bits of SIO wires connect to LIO wires on the left and the other 4 bits connect to LIO wires on the right, transferring a total of 8 bits from each MAT without changing the number of LIO wires. Finally, CSL connections are modified so that 8 bitlines, instead of 4 bitlines, are transferred with a single CSL. As described in Section II-B, CSL connects to 4 bitlines of each and every sub-array within the sub-bank. Hence, instead of naively connecting CSL to twice as many bitlines in each sub-array, we connect CSL to twice as many bitlines in alternating sub-arrays. In other words, CSL connects to twice as many bitlines within each sub-array, but it is associated with only half as many sub-arrays as before. Although this modification does not reduce the number of CSL wire tracks, it halves the number of CSL wires toggled during column operation.

After this simple modification, a row operation selects half of the MATs within a sub-array in half page mode. Only wordlines and sense amplifiers of selected MATs are enabled, fetching half as many bits to the row buffer as conventional DRAM. Also during column operation, 8 bits of data are transferred in and out of each MAT instead of 4 bits. Therefore, our design fetches only half as many bits to the row buffer while maintaining full bandwidth from the bank.

C. Interface Support for Half Page DRAM

We make a small change in the activate command interface protocol to enable half page row activation without needing additional pins or spare command codes. One of the existing pins on the column command array is designated as RFU (Reserved for Future Use), so we use that to indicate whether the column command is actually a dummy command. The dummy command can provide the extra address bit needed to initiate half page row activation, while the actual column operation is not performed. With this additional command, the DRAM stores row address information from the activate command for one clock cycle. Then, the dummy command is issued on the next cycle to initiate a row activation by merging the address provided by the dummy and activate commands. The additional cycle used for the dummy command is not expected to degrade performance much due to frequent bubbles present in the CA bus traffic [16], while row activation is ensured to occur on the next cycle of the activate command.

D. Charge Recycling Refresh (CRR)

The drivers for the sense amplifier power supply, SAP and SAN, are located at the sub-hole which is the cross-section of the wordline drivers and the sense amplifiers [17]. Each SAP and SAN supplies power to half of the sense amplifiers in each MAT by a metal wire [18]. In conventional DRAMs, bitline (BL) and reference bitline (BLB) are equalized by shorting them to each other, and sharing the charge on the lines as shown in Figure 7a. When a cell is accessed, charge sharing between the cell and the BL causes a small perturbation to the charge stored on the BL. The sense amplifier then senses the charge difference between BL and BLB and amplifies them to full digital value using charge supplied by the power supplies SAP and SAN as shown in Figure 7b. The concept of charge

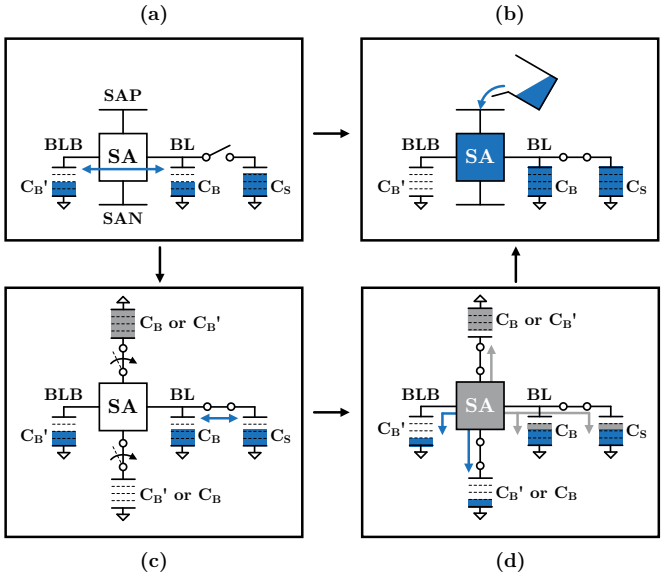


Fig. 7. Charge movements in bitlines for conventional refresh (a→b) and CRR (a→c→d→b): (a) Idle. BL and BLB are equalized, and cell is isolated from the sense amplifier. (b) Amplify. Bitlines and cell are fully restored using the charge supplied by the power supply lines, SAP and SAN. (c) Sensing. The cell's data is sensed by connecting fully restored bitlines of another MAT to the sense amplifier. (d) Charge Recycling. Bitlines of another MAT supply half of the charge needed to fully restore the bitlines.

recycling is to use the fully amplified charge stored on BL and BLB of another MAT as a battery which supply charge to the BL and BLB being sensed. Hence, the sense amplifier begins to sense the data stored on the cell by recycling charge from another MAT's unequalized bitlines as shown in Figure 7c. At the end of this charge recycling process, half of the charge to fully amplify BL and BLB has been supplied from bitlines of another MAT as shown in Figure 7d. BL and BLB are then fully restored by disconnecting the "battery" bitlines, and connecting the sense amplifier to SAP and SAN. It is notable that the BL and BLB lines that were used as the battery will still result in the correct half V_{DD} precharge voltage when they are equalized.

Our new sub-array structure, shown in Figure 5b, makes charge recycling refresh (CRR)[1] [19] feasible in modern DRAM for the first time. Since a neighbor MAT always belongs to a different half page, CRR can be performed on half pages simply by shorting SAP and SAN of one half page to the SAN and SAP of the other as shown in the top side of Figure 8. For better productivity, dummy cells are also placed on the voids caused by adding the switches. Because dummy cells are solely for pattern matching, wordlines of these cells are connected to V_{BBW}, ground voltage of the wordline, and bitlines are connected to V_{BLP}, bitline precharge voltage.

Operation of CRR is shown in the timing diagram on the bottom side of Figure 8. First, the wordlines in an even half page are selected and bitlines associated with the wordlines are

[1]CRR is built on top of the new sub-array structure and it is independent of Half Page DRAM. Hence, CRR does not need additional design changes to transfer twice as much data out from each MAT.

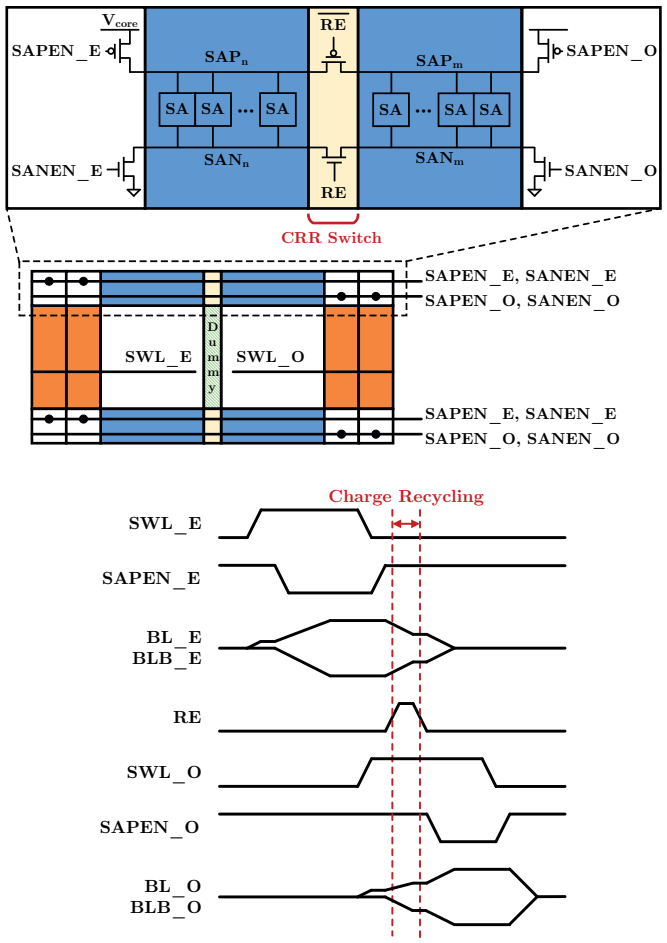
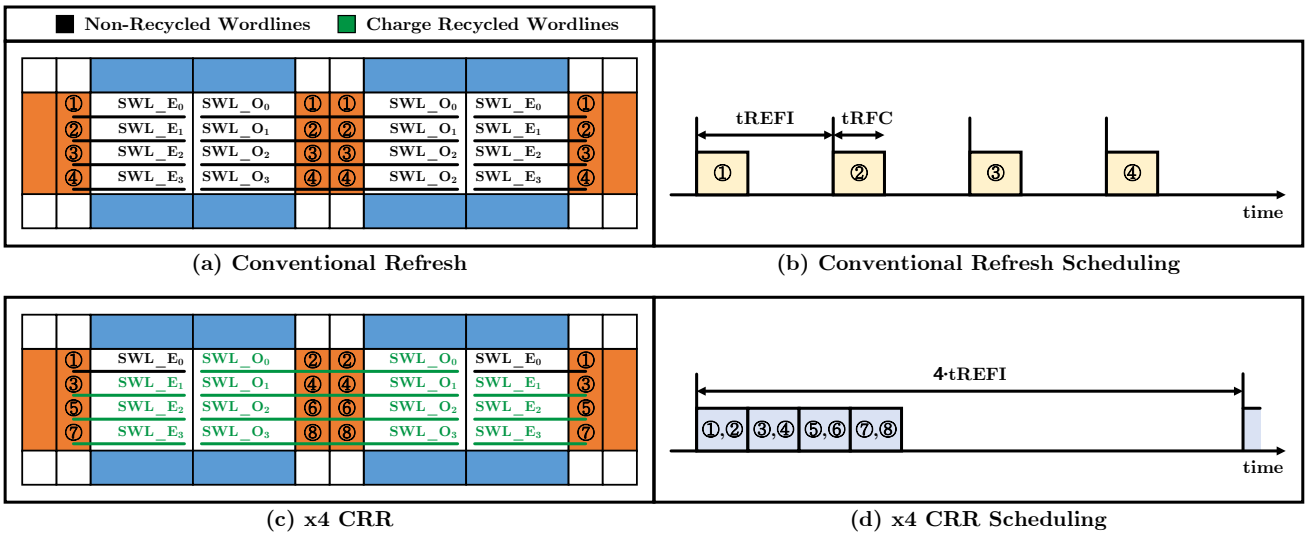


Fig. 8. Design of the proposed charge recycling refresh scheme (top) and its operation (bottom). Switches are added between two adjacent MATs that are on different half pages. Charges are recycled from the bitlines of even half page to odd half page through SAP and SAN wires by turning switches on.

amplified by the sense amplifiers. When data is fully restored, wordlines and sense amplifiers are deselected, but bitlines are not yet precharged. Then, the wordlines in the odd half page are selected, sharing charge between the cells and the bitlines. Charge recycling occurs by enabling *RE*, which shorts SAP_n to SAP_m and SAN_n to SAN_m. Because MAT_n and MAT_m are identical, up to half of the charge stored on the bitlines of MAT_n are transferred to the bitlines of MAT_m using SAP and SAN. After recycling charge, bitlines in the even half page can be precharged and the sense amplifiers of the odd half page can be enabled to supply the rest of the charge to the bitlines using SAP_m and SAN_m.

E. Multiple Row CRR

Ideally, half of the charges on the bitlines are recycled using CRR because the bitline capacitance of two half pages is the same. However, one half page is fully charged using the power supply and only one half of the page is supplied with recycled charge, achieving up to 25% less energy (instead of 50%) to charge and discharge bitlines during refresh than conventional. Because there are multiple rows within a MAT,



(a) Conventional Refresh (b) Conventional Refresh Scheduling

(c) x4 CRR (d) x4 CRR Scheduling

Fig. 9. Comparing refresh sequence and refresh scheduling scheme between conventional and ×4 CRR when four full page rows are refreshed by issuing 4 refresh commands. (a) Order of the rows refreshed in conventional refresh. (b) Conventional refresh scheduling scheme where refreshes are issued periodically every tREFI and each refresh consuming tRFC time to complete refresh. (c) Order of the rows refreshed in ×4 CRR. Charges are recycled to half page rows except for even half page of first refresh. (d) Proposed refresh scheduling scheme where 4 refreshes are issued back-to-back.

we can continue recycling charge to half pages on different rows using CRR, reducing refresh energy even further. We name this feature *multiple row CRR*.

To support selecting two half page rows with different row addresses, we need to slightly change the peripheral circuitry. We propose to store, in each sub-bank, the row address associated with FX and one column address bit that distinguishes FX_E and FX_O for each half page. This allows us to select both FX_E and FX_O lines that are completely different from the row address, which was not possible before. Typically there are 8 FXs in each MWL [6] hence, the proposed multiple row CRR can recycle charge for up to 15 half page rows.

Each half page row has to be refreshed in consecutive order as shown in Figure 9c to enable multiple row CRR. For this purpose, we propose to pull-in multiple refreshes where each refresh is performing CRR as shown in Figure 9d. JEDEC standards allow DDR3 and DDR4 to pull in and postpone up to 8 refreshes [13], [14]. This feature provides an extra memory controller knob to flexibly schedule refreshes based on the workloads. For instance, refreshes can be pulled-in when memory is being used less often, so that future refreshes can be postponed to free up extra memory bandwidth later when it is being used more often. Multiple row CRR can pull-in CRRs to concatenate multiple CRRs back-to-back. Once CRRs are issued consecutively, we can continue recycling charge by not closing the row on the previous refresh cycle until it finishes recycling charge to the row on the following refresh cycle. For auto-refresh, the memory controller can inform the DRAM whether to close the row or not after each CRR by using one address bit as the refresh command is issued. During self-refresh, DRAM can determine by itself when to close a row.

To better illustrate how multiple row CRR works, Figure 9 compares the order of the rows being refreshed and corresponding refresh scheduling for conventional refresh versus proposed multiple row CRR. As an example, we show the case where four full page rows recycle charge for multiple row CRR, which is denoted as ×4 CRR in Figures 9c and 9d. In conventional DRAM, refreshes are performed at full page granularity as shown in Figure 9a. Refreshes are issued periodically every tREFI by the memory controller and each refresh takes time tRFC as shown in Figure 9b. In ×4 CRR, however, refreshes are performed at half page granularity and two half pages are refreshed consecutively in one refresh cycle to recycle charge from even to odd half pages as shown in Figure 9c. Also, four refreshes are issued back-to-back so that charges are recycled to seven half page rows consecutively, saving even more energy than just using CRR or ×1 CRR. Successive refresh(es) are issued after 4·tREFI from the start of ×4 CRR as shown in Figure 9d. This ensures that every cell is refreshed without violating the refresh specification.

The proposed refresh scheduling for multiple row CRR is effective because it provides great flexibility in trading off refresh energy with performance. The memory controller can change the number of CRRs to pull-in on-the-fly to either maximize refresh energy savings or performance, depending on the workload. It is notable that refresh energy is saved even when maximum performance is needed because CRR is used instead of conventional refresh. Moreover, the limit of multiple row CRR matches with the limit of the number of CRRs that can be pulled-in, which allows exploitation of multiple row CRR to its full extent using the proposed refresh scheduling.

IV. Design Analysis

Two new designs—Half Page DRAM and CRR—were proposed in the previous section that exploited half page to reduce row buffer overfetch cost and refresh energy. In this

section, we analyze the potential benefits and cost of their implementation.

A. New Row Parallelism by Half Page DRAM

Two factors degrade performance in Half Page DRAM: (i) a one cycle delay for row activation, and (ii) an additional row activation required when column accesses hit both half pages on the same row back-to-back. However, Half Page DRAM also provides an additional degree of row parallelism, which can potentially improve overall performance even with the above-mentioned performance overheads.

Kim et. al. [20] proposed SALP, which exploits independence between sub-arrays and overlaps accesses to rows in different sub-arrays to reduce the negative impact of bank conflicts. But because row buffers are shared between two adjacent sub-arrays as described in Section II-B, sub-arrays are not completely independent to each other, which limits the operation of SALP. This limitation becomes even severe when row repair is considered because multiple chips in the module have different repair mappings, increasing the possibility of adjacent sub-array access. Unlike sub-arrays, row buffers are not shared between any two half pages within the whole bank. Hence, half page level parallelism can provide row parallelism equivalent to having twice as many banks.

We call the additional row parallelism provided with Half Page DRAM as Half Page Level Parallelism (HPLP)[2]. The precharge time of one half page row is completely overlapped with the activate of any other half page row within a bank, given that both rows are on different half pages. In this work, we assume that tRP is completely negated because at least 2 DRAM clock cycles of gap exists between precharge and the following activate even if tRP is zero. This is because row activating is delayed within the DRAM for one clock cycle in our proposed Half Page DRAM. To simplify the memory controller, additional decision making to avoid bank conflict is done only when a new memory request is available in the transaction queue ready to be issued.

B. Energy Saved Using Half Page DRAM

A 55nm 2Gb DDR3 Rambus model [21] was used to break down row energy for each power supply, V_{DD} and V_{PP}. Most, i.e. 69% of the V_{DD} power supply energy, is used to charge and discharge bitlines. The rest of the energy is dissipated in the periphery circuits to generate and transfer necessary control and address signals to the row decoder. The V_{PP} power supply is used to raise wordlines whose energy we break down further into MWL, FX, and SWL as discussed in Section II-A. Based on the model, MWL consumes 18%, FX consumes 74%, and SWL consumes 8%.

Half Page DRAM reduces row energy by activating only half of the page compared to conventional DRAM. This is done by selecting half as many SWLs and sense amplifiers as before. In our design, MWL is shared with every cell on the full page but FX is connected to only half of the cells to selectively enable half of the SWL. This results in 50% of energy to toggle load and wire of SWL, and 50% of energy to toggle load of FX but not the wire of FX. Overall, Half Page DRAM saves 39% of the V_{PP} power supply energy. Selecting only half of the sense amplifiers reduces energy to charge and discharge bitlines by half, which saves 37% of the V_{DD} power supply energy. In total, Half Page DRAM results in IDD0 of 31.8mA instead of 34mA and IPP0 of 3.6mA instead of 4mA compared to the baseline device described in Section V. Resulting IDD0 assumes IDD3N and IDD2N during active and idle standby respectively.

Column energy is also reduced by using Half Page DRAM because CSL transfers twice as many bitlines than conventional, reducing the energy to toggle CSL by half. This saves 4.5% of the total column energy which results in IDD4R of 100.7mA instead of 104mA and IDD4W of 78.8mA instead of 81mA.

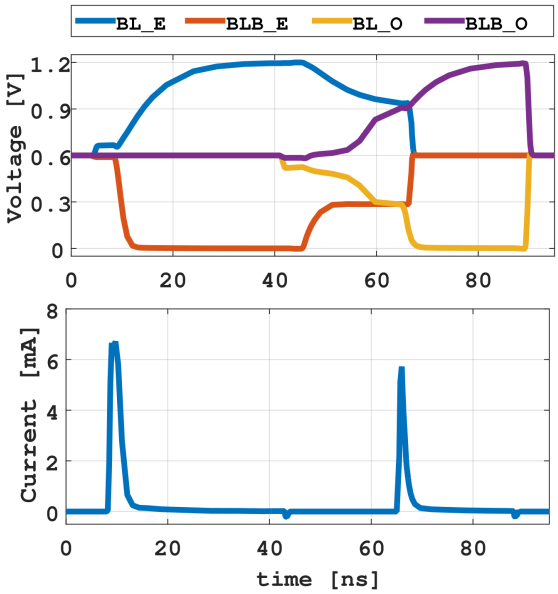


Fig. 10. SPICE simulation of CRR when charge transfer time is 20ns; bitline development (top) and V_{ss} current (bottom). Narrower and shorter current peak shown during amplification of odd half page bitlines indicates that less charge is supplied by the power supply than before.

C. Charge Transfer Time of CRR

The SPICE simulation of CRR shown in Figure 10 used the Predictive Technology Model [22], which best fits DRAM timing constraints for the transistor size on the 55nm Rambus Power Model [21]. We chose the size of the switch that shorts SAP and SAN of two adjacent MATs to be roughly half the size of the SAP and SAN drivers. The size was chosen to recycle half of the charge stored on bitlines from one half page to the other at $tRAS_{min}$. As shown in the top half of Figure 10, charge is recycled from even half page to odd half page, which can then amplify bitlines of the odd half page based on the cell data sensed by the sense amplifier. It is clear

[2]In this work, we only consider avoiding tRP penalty caused by bank conflicts, which is equivalent to SALP-1. Although half page level parallelism can be used to completely overcome the row buffer limitation of SALP-2 and MASA as well, it does not remove the row repair limitation that SALP-2 and MASA have and which affects die yield.

Temp.	10ns	15ns	20ns	30ns	35ns
85°C	33.3%	42.1%	46.3%	49.3%	50.0%
55°C	39.0%	45.6%	48.4%	49.8%	50.0%
25°C	44.0%	48.3%	49.5%	49.9%	50.0%

TABLE I
PERCENTAGE OF CHARGE RECYCLED AS CHARGE TRANSFER TIME AND TEMPERATURE VARY.

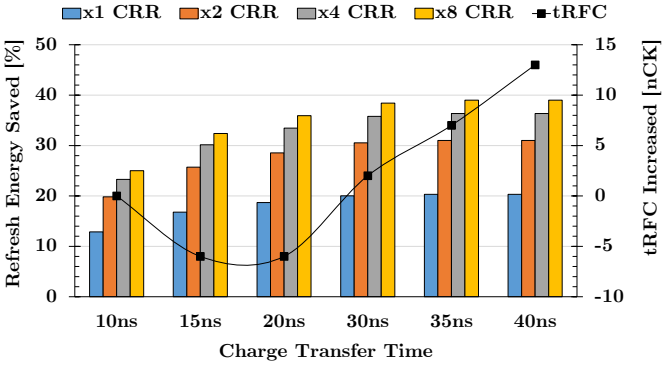


Fig. 11. Refresh energy saved for V_{DD} power supply with CRR at 85°C. tRFC change as charge transfer time varies is also shown.

that, by recycling charge, less energy is dissipated from the power supplies, as shown at the bottom of Figure 10.

Table I shows how much charge is recycled from one half page to the other half page as the charge transfer time and temperature are varied. Because the switch shorting SAP and SAN of two adjacent MATs act as a resistor while transferring charge, more charge is transferred as temperature gets colder. As expected, 50% of the charge is recycled starting around $tRAS_{min}$ which is between 30ns to 35ns. We can also see that even at 85°C, more than 40% of the charge transfer time is spent to recycle the last 7% of the charge.

D. Energy Saved Using CRR

CRR saves refresh energy by reducing the energy to charge and discharge bitlines. This reduces the energy dissipated by the V_{DD} power supply but does not affect the energy dissipated by the V_{PP} power supply. Figure 11 shows the effect of CRR on refresh energy using V_{DD} power supply at 85°C. Even with $\times 1$ CRR and 20ns of charge transfer time, CRR effectively reduces 18.7% of the V_{DD} powered refresh energy. More energy is saved as more charge is recycled, which can be achieved by either increasing the charge transfer time or by increasing the number of rows that recycle charge. Expected refresh energy savings shown in Figure 11 conservatively assumed that any repaired row would result in $\times 1$ CRR for all refreshes being pulled-in during multiple-row CRR. We also assumed 1.56% of the total rows were repaired rows [23], hence for $\times 8$ CRR, 87.5% of the refreshes were considered as $\times 8$ CRR and the rest as $\times 1$ CRR.

Refresh cycle time (tRFC), which is the time that memory is blocked from use during refresh, is also a very important parameter for the refresh operation. tRFC is constrained by the power spent to refresh many cells simultaneously. It is

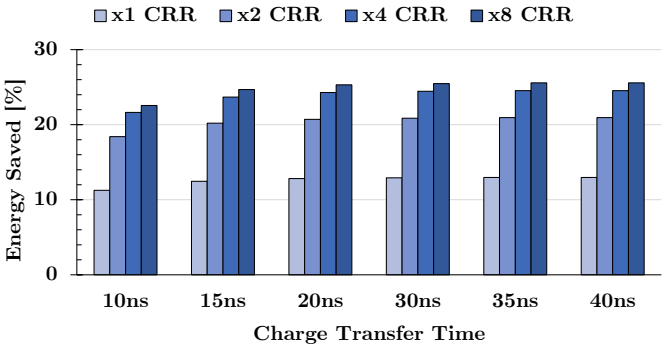


Fig. 12. DRAM standby energy saved when CRR is used during self-refresh at 25°C. Refresh energy portion during self-refresh mode is assumed to be 80% of DRAM energy [24].

set to avoid instantaneous drop in power supply voltage, especially V_{DD}, so that data is fully restored back to the cell by refresh. CRR can potentially reduce tRFC because (i) refresh power is distributed over time by refreshing one half page at a time and (ii) less energy is dissipated to refresh the same amount of cells. Figure 11 shows the change in tRFC compared to conventional as the charge transfer time and reduction rate of V_{DD} power supply refresh energy change. We conservatively assumed 5ns margin for the resulting tRFC to provide sufficient time between control signals added to operate CRR. tRFC is estimated to be reduced by 6 clock cycles[3] for 15ns and 20ns of charge transfer time, while the same tRFC can be used for 10ns of charge transfer time. And the overhead of charge transfer time to tRFC starts to get greater, increasing tRFC as charge transfer time becomes longer than 30ns. Hence, we use 20ns of charge transfer time for the remaining part of the work, which saves 14.9% of V_{DD} and V_{PP} combined refresh energy with 6 clock cycles less tRFC for $\times 1$ CRR. Auto-refresh current (IDD5B) changes from 155mA to 134mA and IPP5 changes from 18mA to 18.3mA than the baseline device described in Section V.

Refresh energy is a significant portion of total energy at low temperatures because static leakage is suppressed as temperature gets colder. Hence, it is not surprising that a separate Micron product line, DDR3L-RS, reduces its self-refresh rate by half at temperatures below 45°C to reduce overall DRAM energy. DDR3L-RS is reported to reduce 40% of standby power at 25°C [24], which means refresh itself consumes roughly 80% of the standby power during self-refresh. Figure 12 shows total DRAM energy saved at 25°C when DRAM is in self-refresh mode. We use the reported energy breakdown for DDR3L-RS self-refresh, which we think is conservative considering the trend in refresh energy described in Section II-D. The proposed refresh scheduling can fully utilize the potential of multiple row CRR in self-refresh mode by using $\times 8$ CRR to refresh cells while in self-

[3]Additional time needed by CRR is only counted for the last row being refreshed and, with 20ns of charge transfer time, it increases tRFC by $20ns + (tRAS - tRCD) = 38.3ns$. As the effect of energy saved and 5ns margin are applied in addition to the previously calculated tRFC, it changes to $38.3ns - (tRFC \cdot EnergySaved) + 5ns = -5.3ns$.

System:	4-core processor, 4 memory channels
Processor:	2.4GHz Nehalem [26]
LLC (L3$):	shared 8MB, 16-way LRU
DRAM Controller:	64/64 entries of read/write queue FR-FCFS and close-page policy
DRAM:	4GB 1Rx8 DDR4-2400 17-17-17 [27]

TABLE II
FULL SYSTEM CONFIGURATION.

	IDD0	IPP0	IDD5B	IPP5
Baseline	34mA	4mA	155mA	18mA
Half Page DRAM	31.8mA	3.6mA	155mA	18mA
CRR	34mA	4mA	134mA	18.3mA

TABLE III
DRAM IDD PARAMETERS.

#	MPKI	Benchmarks
1	Low-Low-Mid-High	gromacs-perlbench-hmmer-soplex
2		h264ref-gromacs-zeusmp-bwaves
3		perlbench-gobmk-bzip2-soplex
4		gobmk-sjeng-wrf-lbm
5		gromacs-gromacs-sphinx3-soplex
6	Low-Mid-Mid-High	sjeng-zeusmp-cactusADM-libquantum
7		gobmk-gcc-dealII-lbm
8		perlbench-hmmer-xalancbmk-milc
9		gromacs-wrf-hmmer-lbm
10		h264ref-dealII-hmmer-libquantum
11	Low-Mid-High-High	gobmk-leslie3d-astar-astar
12		sjeng-gcc-bwaves-mcf
13		perlbench-GemsFDTD-libquantum-mcf
14		gromacs-cactusADM-omnetpp-mcf
15		h264ref-zeusmp-lbm-lbm

TABLE IV
SPEC CPU2006 WORKLOAD SETUP.

refresh and $\times 1$ CRR as we exit from self-refresh. This can provide maximum energy savings while reducing the penalty to recover from self-refresh thanks to the short tRFC of $\times 1$ CRR. Assuming the same 20ns charge transfer time, 25.3% of standby energy is estimated to be saved by using $\times 8$ CRR during self-refresh

Special auto-refresh features such as per-bank refresh [25] and Fine Granularity Refresh (FGR) [14] can also save energy by applying CRR to them. However, tRFC might increase in return, unlike all-bank auto-refresh. This is because additional time spent to recycle charge from one half page to the other are not fully mitigated by the energy being saved, depending on how many cells are being refreshed.

E. Area Overhead

There is no area overhead in our half page sub-array structure, because we just re-organized the sub-array of DDR4's existing $\times 4$ half page architecture without adding any new components. However, new components added to enable two other designs, Half Page DRAM and CRR, that were built on top of our sub-array structure costs small additional die area.

Half Page DRAM needs additional 4 bit SIO/SIOB wires in each sub-array. Additional SIO/SIOB wires are routed on the sense amplifier region and increase the height of the chip. Sense amplifiers are staggered and shared between two adjacent sub-arrays as described in Section II-B. Hence, 64 additional SIO wires are added instead of 128 wires for each sub-bank because there are 32 sub-arrays in each sub-bank [10]. Area overhead caused by additional SIO wires are estimated by first predicting the pitch of SIO wire using the pitch of the CSL. There are 128 CSLs total within a MAT of 512 columns. If F is feature size then bitline pitch is 2F in a $6F^2$ cell [28], and the wire pitch of CSL and SIO is therefore 8F. For the 55nm Rambus Power Model we used, 64 additional SIO wires plus 64 SIOB increases the height of each bank by $56.3\mu m$, which is estimated to be 1.4% area overhead for the whole chip.

CRR increases the width of the chip by adding switches to short SAP and SAN wires of two adjacent MATs. As described in Section IV-C, the size of the switches were chosen to be roughly half of the SAP and SAN drivers. Each switch increases the width by 32F using the size information provided by the 55nm Rambus Power Model that we used and conservative design rules. In each sub-array there are 16 MATs [11], resulting in 8 additional switches per bank. This increases the width of each bank by $14.1\mu m$ and is estimated to be 1.2% area overhead for the whole chip. Another component that we added to enable CRR includes the latches and corresponding wires added in each bank to store the one bit column address bit that selects either even or odd half page, and the 3 bit row address that represents FX. The area overhead caused by these latches is negligible considering that adding many more latches for each sub-array, instead of each bank, was also estimated to be negligibly small in other work using the same model [15], [20].

V. EVALUATION METHODOLOGY

We use USIMM [29] to evaluate the potential advantage of our proposed Half Page DRAM and CRR in a system configured as shown in Table II. An FR-FCFS [30] scheduler with write drain mode is used for the memory controller to maximize DRAM throughput. If not mentioned otherwise, we use close-page policy that aggressively closes the row whenever possible. Power-down is also entered to save standby power when there are no pending memory requests in the memory controller and all of the DRAM banks are precharged. Baseline DRAM timing and IDD parameters are extracted from the data sheet of Samsung K4A4G085WD parts [27]. A 4GB DRAM module is formed using eight 4Gb DDR4-2400 DRAM chips with CL-nRCD-nRP setting of 17-17-17. The same I/O termination suggested by Micron's DDR4 power calculator spreadsheet [31] is used, hence, the termination powers are extracted directly from the spreadsheet. Table III summarizes IDD parameters affected by proposed design schemes as analyzed in Sections IV-B and IV-D.

Proposed designs are evaluated using multi-program workloads composed using SPEC CPU2006 [32] benchmarks as listed in Table IV. Four benchmarks are selected at random from each bin of MPKI (Miss Per Kilo-Instruction) categorized by Low, Mid, and High. We filtered out five benchmarks that showed less than 0.1 MPKI prior to classifying SPEC CPU2006 benchmarks in their respective MPKI bins. The

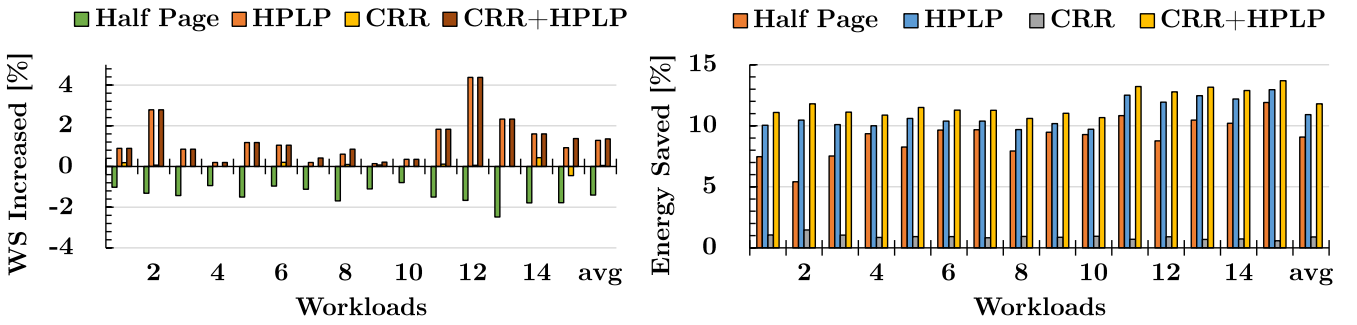


Fig. 13. Weighted Speedup (WS) increased (left) and DRAM energy saved (right) for different combinations of designs: (i) Half Page DRAM, (ii) HPLP, (iii) CRR, and (iv) CRR+HPLP.

DRAM footprint of the workloads is collected for 500 million instructions using Snipersim [33] in a system configured as in Table II. Our workloads utilize 8% to 33% of the peak DRAM bandwidth available, which we believe is representative of real use cases [1]. Finally, the effectiveness of proposed designs are reported using a weighted speedup metric for performance [34] and an energy-per-bit metric for energy.

VI. Results

We evaluate performance and energy impact of proposed Half Page DRAM and CRR in quad-core systems. The left chart in Figure 13 shows weighted speed-up improvements for different combinations of proposed designs. Half Page DRAM using the proposed interface described in Section III-C degrades performance by 1.4% on average and up to 2.5%. As already discussed in Section IV-A, the performance overhead is due to one additional clock cycle needed to activate a row, plus additional cycles spent to activate more rows when both half pages within a row are accessed simultaneously. On average only 0.3% of the row activations access both half pages within a row. The maximum penalty that has to be paid to access another half page on the same row is tRRD + tRCD from the previous half page row activation. However, this penalty can be hidden in part by the column latency of the previous half page. Hence, the one additional cycle spent to activate a row is a major reason for the slight performance degradation of Half Page DRAM. Although not shown, this is why benchmarks with high MPKI of each workload contributes most to weighted speedup change among other benchmarks constituting a workload.

When the memory controller utilizes HPLP, the performance degradation is removed completely and improves the weighted speedup by 1.3% on average. Results correlate well with the ratio of half pages that can be accessed in parallel by HPLP, i.e. half pages on a different row and a different half page than the current half page being accessed. The average ratio of half pages that benefits from HPLP is 11.1%. As explained in Section IV-A, we allowed HPLP only when a memory request that can be parallelized is residing on the transaction queue of the memory controller. Hence, an open-page policy benefits from HPLP more than the close-page policy we used where 1.9x speedup is observed because 2.8x more half-pages benefit from HPLP. Based on the evaluation results, HPLP effectively removes the performance degradation of Half Page DRAM by hiding part of the bank conflict overheads.

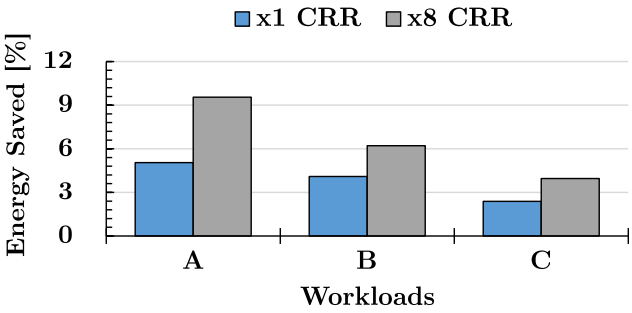


Fig. 14. Comparing DRAM energy saved when either ×1 CRR or ×8 CRR replaces conventional auto-refresh in quad-core systems with 2 channels of 8GB 1Rx8 DDR4 DRAM [35]. Workloads are composed by mixing low MPKI SPEC CPU2006 benchmarks.

CRR alone has negligible effect on performance with only 0.1% average improvement. ×1 CRR with 20ns charge transfer time, used with auto-refresh, saves tRFC by 6 clock cycles for a 1.9% improvement in tRFC. But this is amortized severely because additional accesses are seldom issued in such a short period of time.

Unlike performance, energy is saved regardless of the combinations of proposed designs, as shown in the right-hand chart of Figure 13. Half Page DRAM reduces 37% of V_{DD} power supply row energy and 4.5% of column energy by accessing only half of the page. But background energy is increased by 3% on average compared to the baseline because of the performance overhead that Half Page DRAM introduces. Despite this increment in background energy, 9% of the energy is saved on average because Half Page DRAM saves a significant amount of row energy. HPLP is even more effective in saving energy by reducing background energy in addition to the row and column energy savings already provided by Half Page DRAM alone. On average, 11% of the energy is saved by exploiting half page level parallelism in Half Page DRAM.

Although CRR is most effective in saving energy at cold temperatures and self-refresh, it can also be used for auto-refresh to save refresh energy when DRAM is intensively used. On average 1% of DRAM energy is saved when ×1

CRR is applied during auto-refresh. The energy savings are relatively small considering that 14.9% of refresh energy is saved even with $\times 1$ CRR. But refresh energy in memory intensive workloads averages only 6% of total energy, as shown in Figure 3. This explains the small energy savings achieved by using CRR and at the same time strengthens the need for a scheme that can also be applied for self-refresh as CRR does.

Additional evaluations were done to better show the potential of CRR by using workloads composed only with low MPKI benchmarks, as shown in Figure 14. The average DRAM bandwidth utilization of the three different workloads used is 5.5% of the peak bandwidth where workload A uses the least bandwidth and workload C the most. The workloads were simulated with two different settings, where auto-refreshes are completely replaced by either $\times 1$ CRR or $\times 8$ CRR. We also used 8Gb DDR4 DRAM [35] chips instead of the 4Gb DRAM chips that we used in previous evaluations. When the same analysis discussed in Section IV-D is applied to 8Gb DDR4 DRAM, $\times 1$ CRR saves 19.7% and $\times 8$ CRR saves 37% of V_{DD} power supply refresh energy, both with 30 clock cycles less tRFC. Up to 5% and 10% DRAM energy is saved, both with negligible performance impact when $\times 1$ CRR and $\times 8$ CRR is used as shown in Figure 14. It is notable that we assumed 85°C which is the worst case for CRR and most likely not the temperature for a system having such low bandwidth utilization workloads. As future work, we plan to study the effect of CRR in detail with temperature variation, use of self-refresh, and the refresh scheduling we proposed which dynamically pulls-in refreshes depending on the workload.

VII. Related Work

Row Buffer Overfetch. The Half-DRAM proposed by Zhang et al. [15] re-routed wordlines to select half of the cells located in one MAT and half of the cells located in another MAT, instead of every cell in a single MAT. However, contrary to the assumptions underlying Half-DRAM, sub-wordlines of modern DRAM are routed in a staggered manner as discussed in Section II-A. This makes Half-DRAM possible only with a large area overhead, which is estimated to be up to 12% since it would need to double the sub-wordline drivers. Our design takes both staggered sub-wordlines and data wire hierarchy into account and still achieves 1.4% area overhead.

Refresh Energy. Previous work suggested reducing refresh energy by using a longer refresh interval (tREFI) that can still retain data stored on majority of the cells. The small fraction of weaker cells that can no longer retain data are either not used [36], [37] or refreshed more frequently than the others [38]–[40]. However, long and frequent on-the-fly retention time characterizations of the cells are needed to identify weak cells because of Variable Retention Time (VRT) [41]–[43]. Those proposals are also not applicable to self-refresh where refresh energy is significant [5]. Our proposed CRR, on the other hand, can be used for both auto- and self-refresh and doesn't rely on statistical characteristic of cells. Moreover, CRR is orthogonal to the previous work and can be applied on top of those schemes to save even more energy.

Charge Recycling Refresh. Kawahara et. al. [19] implemented a Charge Recycle Refresh scheme in a test chip that consisted of two small MATs. Charges were recycled between two MATs by shorting power supply lines, similar to our proposed CRR. They also extended their design to recycle charge stored in bitlines of one MAT to bitlines of multiple MATs in round robin order. Although ideally that saves more energy, it increases tRFC significantly and it adds many more switches than our design. Moreover, it is only suitable when there is a small number of MATs where MAT granularity row access is feasible. We instead exploit half page access granularity which is already supported by DDR4 and we allow the memory controller to flexibly schedule CRR so that refresh energy can be saved without degrading performance. Nevertheless, it is encouraging work that provides proof of concept for our proposed CRR.

VIII. Conclusion

We proposed two new designs, Half Page DRAM and CRR, that exploit half page access granularity to reduce the row buffer overfetch cost and refresh energy. Both of the designs are built on top of a new sub-array structure that re-organized the existing half page architecture of DDR4. We also take cost-per-bit optimization of modern DRAM into account to minimize the area overheads of proposed designs. Our evaluations show that Half Page DRAM is an effective solution in reducing row buffer overfetch cost, where both row and column energy is reduced while slightly improving performance. CRR has negligible impact on performance and saves refresh energy for both auto- and self-refresh. More refresh energy can be saved on demand using CRR by pulling-in refreshes to recycle charges to more rows. CRR is even more effective as density increases and technology node scales, which is the trend in DRAM. The proposed designs should provide a new approach to solving issues posed by today's DRAM such as scalability, four bank activation window (tFAW), and row hammer [44].

Acknowledgment

We thank the anonymous reviewers for their helpful feedback. This work was supported in part by C-FAR, one of the six SRC STARnet Centers, sponsored by MARCO and DARPA. Heonjae Ha is supported in part by Kwanjeong Educational Foundation. Shahar Kvatinsky is partially supported by the Viterbi Fellowship in the Technion Computer Engineering Center.

References

[1] K. Malladi, F. Nothaft, K. Periyathambi, B. Lee, C. Kozyrakis, and M. Horowitz, "Towards energy-proportional datacenter memory with mobile DRAM," in *Computer Architecture (ISCA), 2012 39th Annual International Symposium on*, June 2012, pp. 37–48.

[2] D. Kaseridis, J. Stuecheli, and L. K. John, "Minimalist open-page: A DRAM page-mode scheduling policy for the many-core era," in *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO-44. New York, NY, USA: ACM, 2011, pp. 24–35. [Online]. Available: http://doi.acm.org/10.1145/2155620.2155624

[3] J. H. Ahn, J. Leverich, R. Schreiber, and N. Jouppi, "Multicore DIMM: an energy efficient memory module with independently controlled DRAMs," *Computer Architecture Letters*, vol. 8, no. 1, pp. 5–8, Jan 2009.

[4] J. H. Ahn, N. P. Jouppi, C. Kozyrakis, J. Leverich, and R. S. Schreiber, "Improving system energy efficiency with memory rank subsetting," *ACM Trans. Archit. Code Optim.*, vol. 9, no. 1, pp. 4:1–4:28, Mar. 2012. [Online]. Available: http://doi.acm.org/10.1145/2133382.2133386

[5] M. A. Viredaz and D. A. Wallach, "Power evaluation of a handheld computer," *IEEE Micro*, vol. 23, no. 1, pp. 66–74, Jan. 2003. [Online]. Available: http://dx.doi.org/10.1109/MM.2003.1179900

[6] K. Koo, S. Ok, Y. Kang, S. Kim, C. Song, H. Lee, H. Kim, Y. Kim, J. Lee, S. Oak, Y. Lee, J. Lee, J. Lee, H. Lee, J. Jang, J. Jung, B. Choi, Y. Kim, Y. Hur, Y. Kim, B. Chung, and Y. Kim, "A 1.2V 38nm 2.4Gb/s/pin 2Gb DDR4 SDRAM with bank group and half-page architecture," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2012 IEEE International*, Feb 2012, pp. 40–41.

[7] 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 *High Performance Computer Architecture (HPCA2013), 2013 IEEE 19th International Symposium on*. IEEE, 2013, pp. 615–626.

[8] A. N. Udipi, N. Muralimanohar, N. Chatterjee, R. Balasubramonian, A. Davis, and N. P. Jouppi, "Rethinking DRAM design and organization for energy-constrained multi-cores," *ACM SIGARCH Computer Arch. News*, vol. 38, no. 3, pp. 175–186, 2010.

[9] B. Keeth, R. J. Baker, B. Johnson, and F. Lin, *DRAM Circuit Design: Fundamental and High-Speed Topics*, 2nd ed. Wiley-IEEE Press, 2007.

[10] K.-N. Lim, W.-J. Jang, H.-S. Won, K.-Y. Lee, H. Kim, D.-W. Kim, M.-H. Cho, S.-L. Kim, J.-H. Kang, K.-W. Park, and B.-T. Jeong, "A 1.2V 23nm 6F2 4Gb DDR3 SDRAM with local-bitline sense amplifier, hybrid LIO sense amplifier and dummy-less array architecture," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2012 IEEE International*, Feb 2012, pp. 42–44.

[11] Y. Moon, Y.-H. Cho, H.-B. Lee, B.-H. Jeong, S.-H. Hyun, B.-C. Kim, I.-C. Jeong, S.-Y. Seo, J.-H. Shin, S.-W. Choi, H.-S. Song, J.-H. Choi, K.-H. Kyung, Y.-H. Jun, and K. Kim, "1.2V 1.6Gb/s 56nm 6F2 4Gb DDR3 SDRAM with hybrid-I/O sense amplifier and segmented sub-array architecture," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2009 IEEE International*, Feb 2009, pp. 128–129,129a.

[12] Micron, "4Gb DDR4 SDRAM datasheet," https://www.micron.com/products/datasheets/5a6630ef-7182-4c35-85d3-3a61cc797c24.

[13] JEDEC Solid State Technology Association, "DDR3 SDRAM Standard JESD79-3F," July 2012.

[14] ——, "DDR4 SDRAM Standard JESD79-4A," November 2013.

[15] T. Zhang, K. Chen, C. Xu, G. Sun, T. Wang, and Y. Xie, "Half-DRAM: a high-bandwidth and low-power DRAM architecture from the rethinking of fine-grained activation," in *Computer Architecture (ISCA), 2014 ACM/IEEE 41st International Symposium on*. IEEE, 2014, pp. 349–360.

[16] Micron tech. note, "TN-47-10: DDR2 posted CAS# additive latency introduction," http://application-notes.digchip.com/024/24-19971.pdf.

[17] S. Kang, "Semiconductor memory device with sense amplifier driver having multiplied output lines," Feb. 21 2006, uS Patent 7,002,862. [Online]. Available: https://www.google.com/patents/US7002862

[18] J. C. Gealow, "Impact of processing technology on dram sense amplifier design," Ph.D. dissertation, Massachusetts Institute of Technology, June 1990.

[19] T. Kawahara, Y. Kawajiri, M. Horiguchi, T. Akiba, G. Kitsukawa, T. Kure, and M. Aoki, "A charge recycle refresh for Gb-scale DRAM's in file applications," *Solid-State Circuits, IEEE Journal of*, vol. 29, no. 6, pp. 715–722, Jun 1994.

[20] Y. Kim, V. Seshadri, D. Lee, J. Liu, and O. Mutlu, "A case for exploiting subarray-level parallelism (SALP) in DRAM," *SIGARCH Comput. Archit. News*, vol. 40, no. 3, pp. 368–379, Jun. 2012. [Online]. Available: http://doi.acm.org/10.1145/2366231.2337202

[21] T. Vogelsang, "Understanding the energy consumption of dynamic random access memories," in *Microarchitecture (MICRO), 2010 43rd Annual IEEE/ACM Int'l Symposium on*, Dec 2010, pp. 363–374.

[22] W. Zhao and Y. Cao, "New generation of predictive technology model for sub-45 nm early design exploration," *Electron Devices, IEEE Transactions on*, vol. 53, no. 11, pp. 2816–2823, 2006.

[23] Y. Choi, "Under the hood: DRAM architectures: 8F2 vs. 6F2," http://www.eetimes.com/document.asp?doc_id=1281208.

[24] Micron, "DDR3L-RS: Reducing DRAM power consumption in standby," https://www.micron.com/about/blogs/2013/january/ddr3l-rs-reducing-dram-power-consumption-in-standby.

[25] JEDEC Solid State Technology Association, "LPDDR4 SDRAM Standard JESD209-4A," November 2015.

[26] R. Singhal, "Inside Intel next generation Nehalem microarchitecture," in *Hot Chips*, vol. 20, 2008.

[27] Samsung, "4Gb D-die DDR4 SDRAM datasheet rev. 1.7," http://www.samsung.com/semiconductor/global/file/product/2016/03/DS_K4A4G085WD-B_Rev17-0.pdf, Dec. 2015.

[28] H. Oh, J.-Y. Kim, J. Kim, S. Park, D. Kim, S. Kim, D. Woo, Y. Lee, G. Ha, J. Park, N. Kang, H. Kim, Y. Hwang, B. Kim, D. Kim, Y. Cho, J. Choi, B. Lee, S. Kim, M. Cho, Y. Kim, J. Choi, D. Shin, M. Shim, W. Choi, G. Lee, Y. Park, W. Lee, and B.-I. Ryu, "High-density low-power-operating DRAM device adopting 6F2 cell scheme with novel S-RCAT structure on 80nm feature size and beyond," in *Solid-State Device Research Conference, 2005. ESSDERC 2005. Proceedings of 35th European*, Sept 2005, pp. 177–180.

[29] 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, Tech. Rep*, 2012.

[30] S. Rixner, W. Dally, U. Kapasi, P. Mattson, and J. Owens, "Memory access scheduling," in *Computer Architecture, 2000. Proceedings of the 27th International Symposium on*, June 2000, pp. 128–138.

[31] Micron Technology, Inc., "DDR4_Power_Calc.XLSM," https://www.micron.com/~/media/documents/products/power-calculator/ddr4_power_calc.xlsm, Version 1, Oct. 1, 2014.

[32] J. L. Henning, "SPEC CPU2006 benchmark descriptions," *ACM SIGARCH Computer Architecture News*, vol. 34, no. 4, pp. 1–17, 2006.

[33] T. E. Carlson, W. Heirman, S. Eyerman, I. Hur, and L. Eeckhout, "An evaluation of high-level mechanistic core models," *ACM Transactions on Architecture and Code Optimization (TACO)*, 2014.

[34] A. Snavely and D. M. Tullsen, "Symbiotic jobscheduling for a simultaneous multithreaded processor," in *Proceedings of the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS IX. New York, NY, USA: ACM, 2000, pp. 234–244. [Online]. Available: http://doi.acm.org/10.1145/378993.379244

[35] Samsung, "8Gb B-die DDR4 SDRAM datasheet rev. 1.11," http://www.samsung.com/semiconductor/global/file/product/2016/03/DS_K4A8G085WB-B_Rev16-0.pdf, Dec. 2015.

[36] R. Venkatesan, S. Herr, and E. Rotenberg, "Retention-aware placement in DRAM (RAPID): software methods for quasi-non-volatile DRAM," in *High-Performance Computer Architecture, 2006. The Twelfth International Symposium on*, Feb 2006, pp. 155–165.

[37] S. Baek, S. Cho, and R. Melhem, "Refresh now and then," *Computers, IEEE Transactions on*, vol. 63, no. 12, pp. 3114–3126, Dec 2014.

[38] J. Liu, B. Jaiyen, R. Veras, and O. Mutlu, "RAIDR: Retention-aware intelligent DRAM refresh," in *Computer Architecture (ISCA), 2012 39th Annual International Symposium on*, June 2012, pp. 1–12.

[39] Y.-H. Gong and S. Chung, "Exploiting refresh effect of DRAM read operations: A practical approach to low-power refresh," *Computers, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2015.

[40] I. Bhati, Z. Chishti, S.-L. Lu, and B. Jacob, "Flexible auto-refresh: Enabling scalable and energy-efficient DRAM refresh reductions," in *Computer Architecture (ISCA), 2015 ACM/IEEE 42nd Annual International Symposium on*, June 2015, pp. 235–246.

[41] D. Yaney, C. Lu, R. Kohler, M. Kelly, and J. Nelson, "A meta-stable leakage phenomenon in DRAM charge storage - variable hold time," in *Electron Devices Meeting, 1987 International*, Dec 1987, pp. 336–339.

[42] P. Restle, J. Park, and B. Lloyd, "DRAM variable retention time," in *Electron Devices Meeting, 1992. IEDM '92. Technical Digest., International*, Dec 1992, pp. 807–810.

[43] M. Qureshi, D.-H. Kim, S. Khan, P. Nair, and O. Mutlu, "AVATAR: A variable-retention-time (VRT) aware refresh for DRAM systems," in *Dependable Systems and Networks (DSN), 2015 45th Annual IEEE/IFIP International Conference on*, June 2015, pp. 427–437.

[44] Y. Kim, R. Daly, J. Kim, C. Fallin, J. H. Lee, D. Lee, C. Wilkerson, K. Lai, and O. Mutlu, "Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors," in *Proceeding of the 41st annual international symposium on Computer architecuture*. IEEE Press, 2014, pp. 361–372.