Reducing Read Latency of Phase Change Memory via Early Read and Turbo Read

Prashant J. Nair[†] Chiachen Chou[†] Bipin Rajendran[‡] Moinuddin K. Qureshi[†]

[†]*School of Electrical and Computer Engineering*
Georgia Institute of Technology
{pnair6,cchou34,moin}@ece.gatech.edu

[‡]*Department of Electrical Engineering*
Indian Institute of Technology, Bombay
bipin@ee.iitb.ac.in

***Abstract*—Phase Change Memory (PCM) is an emerging memory technology that can enable scalable high-density main memory systems. Unfortunately, PCM has higher read latency than DRAM, resulting in lower system performance. This paper investigates architectural techniques to improve the read latency of PCM. We observe that there is a wide distribution in cell resistance in both the SET state and the RESET state, and that the read latency of PCM is designed conservatively to handle the worst case cell. If PCM sensing can be tuned to exploit the variability in cell resistance, then we can get reduced read latency. We propose two schemes to enable better-than-worst-case read latency for PCM systems.**

Our first proposal, *Early Read*, reads the data earlier than the specified time period. Our key observation that Early Read causes only unidirectional errors (SET being read as RESET) allows us to efficiently detect data errors using Berger codes. In the uncommon case that Early Read causes data error(s), we simply retry the read operation with original latency. Our evaluations show that Early Read can reduce the read latency by 25% while incurring a storage overhead of only 10 bits per 64 byte line. Our second proposal, *Turbo Read*, reduces the sensing time for read operations by pumping higher current, at the expense of accidentally switching the PCM cell with small probability during the read operation. We analyze Error Correction Codes (ECC) and Probabilistic Row Scrubbing (PRS) for maintaining data integrity under Turbo Read. We show that a combination of Early Read and Turbo Read can reduce the PCM read latency by 30%, improve the system performance by 21%, and reduce the Energy Delay Product (EDP) by 28%, while requiring minimal changes to the memory system.

***Keywords*—Phase Change Memory, Read Latency, Error Detecting Codes, Berger Codes, ECC, Reliability, Read Disturbance**

I. Introduction

Phase Change Memory (PCM) is one of the leading emerging technologies [1, 2, 3] that can be used for scaling capacity for future memory systems. While PCM has the advantage of better scalability and non volatility, it has the disadvantage of higher read latency (compared to DRAM), higher write latency and write power, and limited write endurance. A significant research effort has gone in addressing the limited endurance of PCM using either wear leveling techniques [4, 5] or efficient sparing of worn out cells [6, 7, 8]. Furthermore, recent research has also addressed the problem of long write latency [9, 10] and limited write bandwidth [11, 12]. However, there is not as much effort in the architectural domain to address the higher read latency of PCM. Unfortunately, performance impact due to higher read latency of PCM continues to be a critical obstacle for deployment of PCM as main memory. One potential solution to address the high read latency of PCM is to have a *Hybrid Memory System* [2], which combines PCM with a DRAM buffer. However, even with a hybrid memory system, the higher read latency of PCM continues to be a problem for accesses that go to main memory. Our studies show that there are significant gains possible if we address the problem of higher read latency of PCM.

Data is stored in PCM in the form of resistance. PCM cells can be in two states: crystalline (SET) and amorphous (RESET). Read and write operations in PCM are performed by passing current. Figure 1(a) shows the read and write pulses for operating a PCM cell. For converting a cell to the SET state, a current pulse that heats the cell above crystallization point in applied for a long time. For RESET, a narrower pulse with higher current is applied. For reading, the voltage (V$_{rd}$) is kept low enough compared to the SET voltage, and it takes a time duration of T$_{rd}$ to sense the cell resistance. Given that PCM cells have a variation in the cell resistance in either of the two states (as shown in Figure 1(b)), the sensing is done at a point that is intermediate between the highest resistance SET cell (cell A) and the lowest resistance RESET cell (cell B). Sensing is typically done by raising the bit-line voltage to V_{rd} and discharging it through the cell to a sense amplifier, where it is compared with a reference voltage (V$_{ref}$) [13, 14].

The time for detection (T$_{sense}$) is higher for higher resistance state, as it takes longer to discharge. To reduce the sensing time, the reference resistance (R_{ref} (corresponding to the reference voltage) is kept close to cell A [15] . To avoid data sensing errors, the time for sensing the PCM cell gets determined by the worst-case cell resistance in the SET state. Instead of designing to handle the worst-case cell, if we could exploit the variability in PCM devices, then we can reduce the its read latency in common case and also enable efficient scaling. However, to maintain reliable operation, we need to equip the system to handle occasionally occurring data errors. We propose two designs to reduce the read latency of PCM.

Our first proposal, *Early Read*, enables the sensing circuitry of PCM to latch the output of the sense amplifier before the specified sensing time (T$_{rd}$). While Early Read reduces the sensing time for common case cells, it can cause a small fraction of cells to give data errors. Typical memory system are designed such that the probability of a failure is negligibly small (less than 10^{-15} per read operation [16])[1]. However, for Early Read, R$_{ref}$ is modulated to lower sensing latency at the expense of higher error rates from the sensing circuitry.

[1] The BER target of 10^{-15} is dictated primarily by the link error rate, irrespective of the memory technology (PCM or DRAM) [16]. In our studies, we use a default target BER of 10^{-16} for the PCM cells. Our proposed solutions become even more effective when the system is designed to target much lower BER (we analyze the impact of varying BER in Section V-F)

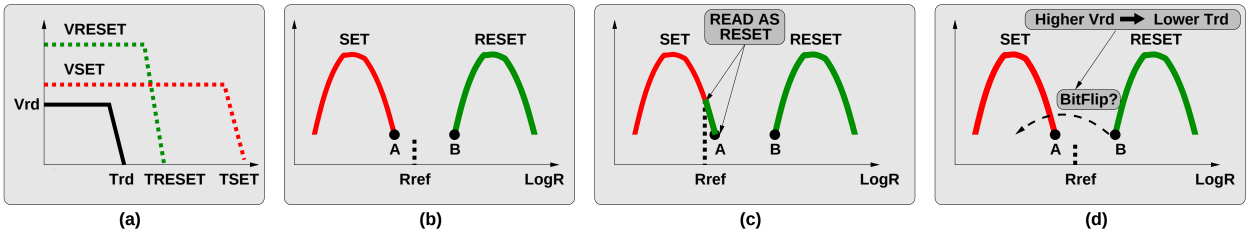

Fig. 1. (a) Timing and voltage for read and write operations (b) Variation in cell resistance for SET and RESET (c) Early Read lowers the reference resistance (R_{ref}) at the expense of some SET cells being read as RESET (d) Turbo Read increases V_{rd} to lower T_{rd}, at the expense of accidentally flipping some RESET cells into SET state.

Our key insight is that Early Read causes only unidirectional errors (SET classified as RESET), as shown in Figure 1(c). To efficiently detect the errors that happen due to Early Read, we equip each line with Berger codes [17], which can detect all unidirectional errors in the line while incurring only 10 bits (for a cache line of 64 bytes). In the uncommon case that Early Read results in a data error, Berger code can detect the error, and the system retries the read with the normal latency. We show that lowering the sensing time by 25% increases the Bit Error Rate (BER) to 10^{-5} and causes probability of retry to still be only 0.5%. Thus, Early Read obtains a low read latency in common case and guarantees correctness in the uncommon case of data error. Our evaluations show that Early Read provides an average speedup of 17%.

Our second proposal, *Turbo Read*, targets the sensing voltage (V_{rd}) as a means to improving read latency. Since writes to PCM system are performed by supplying high current, increasing the read voltage (and thus read current) can cause accidental writes to some cells that are being read, a phenomenon called as *read-disturb* [18, 19]. To avoid read disturb, V_{rd} is set conservatively so that the episode of accidentally flipping the cell while reading happens with a negligibly small probability. Turbo Read increases the sensing voltage so as to reduce the sensing time, and mitigate read disturb faults using error-correcting codes. We show that if the line is provisioned with double error-correcting (ECC-2) code, it can tolerate a BER of 10^{-9} due to read disturb. Given that Turbo Read provides probabilistic guarantees on system reliability, it can also be combined with Early Read (without the need to rely on Berger codes or a retry mechanism) as long as the resulting scheme has an error rate that can be handled with ECC-2. We show that Turbo Read when combined with Early Read can reduce the read latency of PCM by 30%, without the need for the PCM devices to support bimodal read latency (short and normal). Our results for a baseline 8-core system with a 128 MB L4 cache shows that combining Early Read and Turbo Read improves, on average, performance by 21% and Energy Delay Product (EDP) by 28% with an implementation that uses a relatively small hardware budget.

Although the ECC code provisioned with the line is effective at correcting errors due to read disturb, it does so only when the line is accessed the next time. Unfortunately, a read request fetches multiple lines into the PCM row buffer and not all the lines in a row buffer get accessed. A line neighboring to the requested line in the same row can still experience a read disturb due to Turbo Read. If this neighboring line is not read for a long time, then the faults due to read disturb can accumulate over time and can exceed the correction capability of the ECC code associated with that line. We show that such patterns can occur in malicious code or badly written code. To protect the memory from such silent accumulation of read-disturb errors, we propose *Probabilistic Row Scrub (PRS)* that can scrub the referenced row with a small probability. We show that PRS is highly effective at avoiding silent accumulation of faults under Turbo Read.

Furthermore, to handle drift, PCM systems may be designed conservatively to target a much lower BER. We show that our proposal is even more effective in such scenarios, as the margin for drift is provisioned for worst-case cells, and our solutions can exploit this source of cell variability as well.

II. Background and Motivation

We describe the organization of a PCM bank, then discuss the process of a typical read operation in memory, and how the variation in cell characteristics affects the PCM read latency.

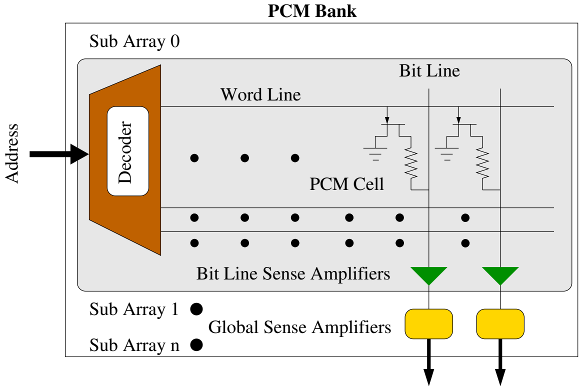

Fig. 2. Overview of a PCM Bank which is composed of subarrays, local and global sense amplifiers. These amplifiers detect values in PCM Cells.

A. Overview of PCM Organization

A PCM module is composed of independent banks, as shown in Figure 2. When a read request arrives into a PCM module, it is routed into the appropriate PCM bank according to its address [14]. The read circuitry precharges the bit line and activates the word-line associated with the resistive PCM element (Germanium-Antimony-Tellurium or GST) [20]. In a two level cell based PCM, the bit-line sense amplifier converges into a digital value of either ‘1’ (low resistance SET state) or ‘0’(high resistance RESET state) depending on the value of GST resistance [13]. To reduce bit-line capacitance, PCM banks are divided into sub-arrays that have their independent sense amplifiers [21]. The bit line sense amplifiers convert the analog PCM cell data into digital data and drive global sense amplifiers for the bank. The global sense amplifiers boost the voltage and drive the data to the I/O pads. The data is transferred from the memory modules to the processor over a bus-based interface such as DDR or LPDDR.

B. Overview of PCM Read Operation

The circuitry and conditions of basic read operation for a cross-bar array PCM cell is chosen so as to satisfy two main constraints - (a) the state of the cell should be read reliably and accurately, while expending the least amount of energy, and as quickly as possible; and (b) the process of reading the cell should itself not alter the state of the cell. Often, these requirements leave a very constrained space in order to design a read circuit for a PCM cell in a large cross-bar array. We describe the voltage based sensing, which is commonly used in many industry prototype chips [22, 23] [2].

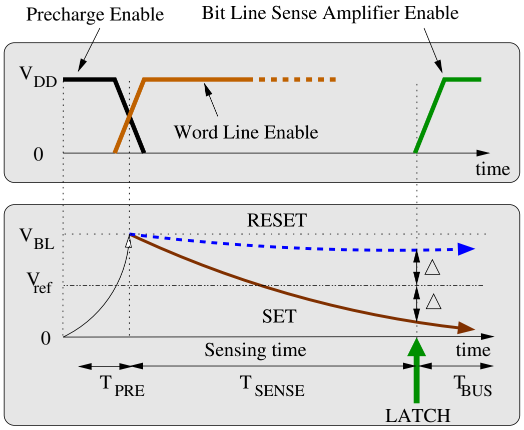

Fig. 3. Timing components for the read operation in a PCM system.

Sensing Time Dominates Read Latency: Figure 3 shows the latency components of a PCM read operation. It consists of three parts: the precharge time (T_{PRE}), the sensing time (T_{sense}), and the data transfer on the bus (T_{BUS}). To read data, the bit-lines are pre-charged to V_{BL}. To read data from a cell, the corresponding word-line is activated. Depending on the value of GST resistance, the voltage in the bit-line will drop. For SET state, due to lower resistance, the voltage will drop faster than the high resistance RESET state. After time T_{sense}, the PCM sense amplifiers compare the bit-line voltage with a reference voltage (V_{ref}). If the voltage in the bit-line is below V_{ref}, the bit-line sense amplifier will provide a '0', otherwise a '1' [22]. To account for device variations, drift and V_{ref} variations, the value of T_{sense} is set conservatively [15][3]. To prevent read-disturb faults, V_{BL} is set conservatively [18, 19]. Overall, the sensing time dominates the read latency of PCM. The time to precharge takes about 5-7ns [3], whereas the time to transfer the cache line on bus takes 4 cycles on a DDR interface (e.g. if the bus operates at 800MHz [22], T_{BUS} is 5ns). The sensing time, however, takes several tens of nanoseconds, and is primarily responsible for the high latency of PCM.

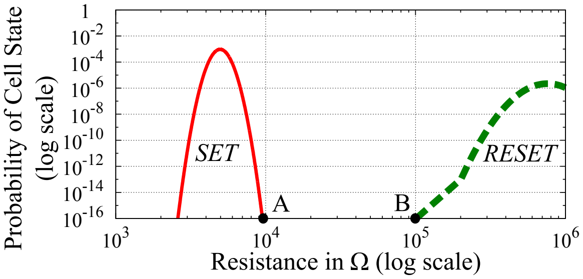

Fig. 4. PCM cells show wide distribution of resistance (derived from cumulative distributions in [25]).

C. Not All PCM Cells Are Created Equal

For reliable differentiation between SET and RESET, there is an order of magnitude difference between the SET resistance (R_{SET}) and RESET resistance (R_{RESET}). However, there is significant variation in the resistance characteristics of different cells, for both states. And the sensing time of the PCM devices are designed to handle the worst-case cell behavior to avoid any sensing related data errors. Prior studies have shown that R_{SET} and R_{RESET} for PCM cells tend to follow a log normal distribution [13, 25]. Figure 4 shows the probability of resistance for a PCM cell in SET and RESET states, based on the PCM characterization data [25]. R_{SET} varies from 1KΩ to 10KΩ, following a distribution (ln$N$[5KΩ,0.0365KΩ]). Whereas, R_{RESET} varies from 100KΩ and can have values as high as 2MΩ following a distribution (ln$N$[800KΩ,0.0995KΩ]).

D. Goal: Better-Than-Worst-Case Read Latency

The sensing time primarily gets determined by the worst-case cell in the SET region (cell A in Figure 4). The sensing voltage gets determined by the most vulnerability cell in RESET state (cell B in Figure 4) to avoid read disturb failures. Rather than making every PCM read operation incur a high latency, just to handle the worst-case cell, we instead want an architecture that can provide low read latency in typical cases and can handle the occasional data errors that may happen with such a design. This paper investigates two Better-Than-Worst-Case (BTWC) [26] designs for read operations in PCM. The first targets the sensing time directly, the second targets sensing voltage as a means to reduce the sensing time. We discuss our methodology before describing our proposals.

III. Experimental Methodology

A. Configuration

We use a detailed memory system simulator for our studies. Table I describes the parameters used in the baseline system. Our baseline has an 8-core processor chip, and that is equipped with a large external L4 cache to implement a hybrid memory system. The L4 cache is 128MB in size, and the cache capacity is partitioned equally between all the cores. The linesize of all cache units is 64 bytes. The main memory system is made of PCM and consists of four channels.

Write requests arrive at the PCM memory only on eviction from the L4 cache. Both read and write requests are at the granularity of cache line, and are serviced by one of the PCM banks, based on the address of the line. Each bank has a

[2] Another sensing mode called current mode sensing can also be used to detect the value in PCM cell. Unfortunately, current mode sensing uses complex circuitry which tends to be intolerant to device variations and is useful only for low density PCM devices [24]. As voltage mode sensing is tolerant to variations, industrial prototypes of high density PCM devices which are used for main memory systems tend to use voltage mode sensing [22, 23].

[3] Drift causes the resistance of PCM cells to increase over time. The sensing time of the PCM array must be increased to compensate for the drift of the worst-case cell in the array. Our default studies conservatively assume no timing margins for drift. We show, in Section V-G, that our proposal becomes even more effective when the margins for drift are taken into account

separate 8-entry read queue (RDQ) and 32-entry write queue (WRQ) that queues all pending requests. The memory system employs a read priority scheduling. We incorporate Adaptive Write Cancellation (AWC) [9] to tolerate the high write latency of PCM. The read latency is assumed to be 80 ns, majority of which (69ns) is consumed by the sensing time [22].

TABLE I. BASELINE CONFIGURATION

Processors	
Number of cores	8 cores, each 4-wide 3GHz
L1/L2/L3 (private)	32KB/256KB/1MB (8-way each)
L4 Cache	
Size	128MB (16MB per core)
Latency	15ns (45 cycles)
Phase Change Memory	
Capacity/Channels	4 channels, each with 8GB DIMM = 32 GB total
Ranks/Banks	1 rank per channel, 8 banks per rank
Read Queue	8-entry/bank (256 total) [9]
Write Queue	32-entry/bank (1024 total) [9]
Read Latency	80 ns (6ns t_{PRE} + 69ns t_{SENSE} + 5ns t_{BUS}) [22]
Write Latency	250ns [22]
Bus (per Channel)	LPDDR, 64-bit (+8-bit for ECC), 800MHz [22]
Scheduling	Read Priority, Adaptive Write Cancellation [9]

B. Workloads

We use a representative slice of 1 billion instructions for each benchmark from the SPEC2006 suite. We perform evaluations by executing the benchmark in rate mode, where all the eight cores execute the same benchmark. Given our study is about memory system, workloads that spend a negligible amount of time in memory are not meaningful. So, we perform detailed study only on the 14 benchmarks that have an L4 misses Per Thousand Instructions (MPKI) of at least 1. Table II shows, for our workloads, the read (MPKI) and the Write Back per Thousand Instructions (WBPKI), both out of the L4 cache. We use an suffix *"_r"* with the benchmark to indicate rate mode. We also use three multiprogrammed workloads: mix_H (8 high intensity workloads, mcf to GemsFDTD), mix_M (8 medium intensity workloads, soplex to zeusmp) and mix_L (8 low intensity workloads, omnetpp to xalancbmk). We perform timing simulation till all the benchmarks in the workload finish execution and measure *Weighted Speedup (WS)*. We report normalized performance (Speedup) as the ratio of WS of the given configuration to the WS of the baseline.

TABLE II. WORKLOAD CHARACTERISTICS.

Workload	L4 Read Miss (MPKI)	L4 WriteBack (WBPKI)
mcf_r	36.9	6.1
libquantum_r	25.4	2.7
milc_r	24.1	9.4
soplex_r	23.5	3.4
bwaves_r	17.2	1.4
lbm_r	16.0	7.8
omnetpp_r	9.5	3.3
GemsFDTD_r	8.4	4.6
gcc_r	5.3	0.7
leslie3d_r	5.1	2.0
zeusmp_r	4.0	1.5
wrf_r	3.2	1.2
cactusADM_r	3.1	0.4
xalancbmk_r	1.1	0.6
mix_H (top 8)	20.1	4.83
mix_M (mid 8)	11.1	3.08
mix_L (bottom 8)	4.96	1.78

IV. EARLY READ

Early Read exploits the variability in the resistance of cells (particularly in SET state) and reduces the sensing time by tuning the sensing circuitry for a cell that has a resistance level well below the highest resistance cell in the SET region. The reduced target resistance decreases the Resistance-Capacitance (RC) time constant for the discharging circuit and reduces the sensing time. This section explains how the variability of PCM cells can be used to determine a reduced sensing time, the architecture support to enable Early Read, and evaluations showing the effectiveness of Early Read in reducing the PCM read latency and the system execution time.

A. Exploiting Cell Variability

The basic idea behind Early Read is to enable sense amplifier to latch the data early thereby reducing the sensing time (T_{sense}), at the expense of causing a few cells in the SET state to be classified as RESET. The tuning of the sensing circuitry with Early Read can be envisioned as a reduction in the value from sensing resistance R_{sense} to a lower value, say R_{sense_ER}. By reducing the value of R_{sense} to R_{sense_ER}, a few cells in the SET have a small probability to be higher than R_{sense_ER} and get classified as RESET. Figure 5 shows the effect reducing R_{sense} on cells in the SET state based on the data obtained from characterization of PCM devices [25]. The value of R_{sense} reduces from 10KΩ (point A) to R_{sense_ER} of 7KΩ (point A*) and all cells between points A* and A become vulnerable to be incorrectly read as RESET, causing data error.

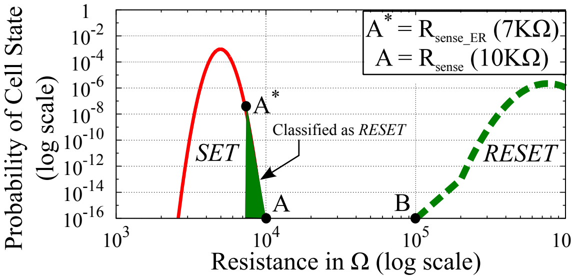

Fig. 5. Early Read reduces the sensing resistance R_{sense} from worst-case cell in SET region (cell A) to a better-than-worst-case cell (cell A*).

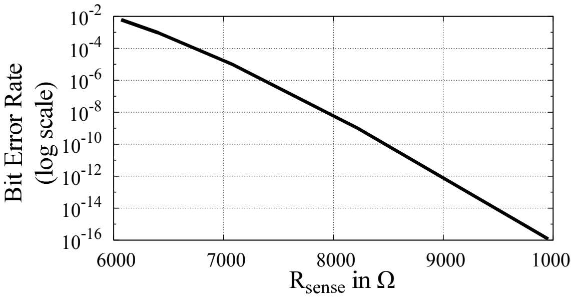

Fig. 6. BER increases exponentially from 10^{-16} to 10^{-5} when R_{sense} is reduces from 10KΩ to 7KΩ.

Figure 6 shows the expected Bit Error Rate (BER) with Early Read as the value of the sensing resistance R_{sense} is changed. This data is derived from Figure 5, and helps us determine the operating R_{sense} for Early Read to get the desired latency reduction at the expense of tolerable BER. For example, for the baseline system R$_{sense}$ is 10KΩ, which means it has negligibly low probability of error. For Early Read, we can select a R$_{sense}$of 7KΩ, resulting in a BER of 10^{-5}. At this BER, we expect one error every 200 read operations.

For a system with Early Read, there must be a provision to tolerate occasional errors. One option is to simply provision the system with Error Correction Codes (ECC) to fix the errors that happen due to Early Read. However, to retain the effectiveness of Early Read, the system needs to be able to tolerate a larger BER (in the regime of several parts per million), and this requires strong ECC code (that can correct almost 6 or more bits per line). Not only are such strong ECC codes expensive in terms of storage but also requires significant latency and complexity. Alternatively, we could go for weaker ECC codes (such as SECDED or DECTED), but those codes can only tolerate a low BER rate (approximately 10^{-9}), making the operation point of the Early Read scheme more conservative and thus less effective. Ideally, we would like to detect any number of errors in the data block, while consuming very little storage overhead and latency.

A key insight that makes our proposed implementation of Early Read reliable and practical is the observation that (a) Latching the output of the sense amplifier early does not affect the state of the PCM cells and (b) Such early latching results in only unidirectional errors (SET read as RESET). We can use a particular type of code, called Berger Code [17] that can efficiently detect all unidirectional errors in a data block while consuming very little storage overhead and complexity. We describe the operation of Berger Codes next.

B. Berger Codes for Error Detection

Berger code is a type of unidirectional error detection code. The check bits of Berger codes are computed by summing all the ones and storing that inverted sum in the check bits. The Berger codes use K check bits for D data bits. The value of K and D follow the relation described by equation (1) [27].

$$K = \log_2(D + 1) \tag{1}$$

Let us assume our two level PCM system classifies SET as logic 1 and RESET as logic 0. Therefore, errors due to Early Read can result in a logical 1 being read as a logical 0. So, to detect such unidirectional errors going from 1 to 0, the check bits of the Berger code can be obtained by counting the number of 1s in the data line and inverting the sum. For a cache line, D is 512b (64Bytes) and the value of K is only 10 bits. When the cache line is written to memory, the check bits are computed by adding all the 1s in the data bits. The computed value is of the sum is inverted, and this value is stored along with the data. When reading from memory, the number of 1s in the data bits is recomputed and its value is compared with the inverted value of check bits. A mismatch signifies that one (or more) unidirectional errors. Figure 7 shows the encoding and detection of Berger code.

Berger Code can detect any number of unidirectional errors that happen either in the data block or the check bits or both. Let us consider the three cases. First, the errors only happen in the data block. Then, the number of ones computed from the data block will be less than the ones reported by the check bits. Second, lets consider that the error only happens in check bits. Then the number of ones in the data block will be lower than the number of ones reported by the check bits. Third, if error happens both in the data block and the check bits, then the sum of ones in the data block can only go down, whereas the sum of the ones reported by the check bits can only go up (as we store the inverted value of sum, the inverted sum only goes down, which means the actual value of stored checked sum can only go up). Thus, unidirectional errors are guaranteed to get detected with Berger codes[4].

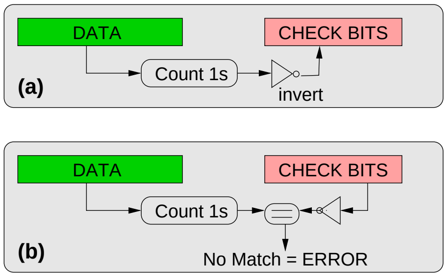

Fig. 7. Berger codes: (a) Encoding the check bits (b) Error Detection.

The storage complexity of Berger code is simply 10 bits per cache line. We assume that each line is equipped with extra bits to store the check bits for Berger code. Note that, PCM memories are likely to be provisioned with error tolerance capability, such as Error Correcting Pointers (ECP) [6] to tolerate endurance related faults anyway. Prior proposals have considered an architecture similar to SECDED DIMM for PCM, whereby the extra 64 bits per 64 byte line are used to correct up-to 6 endurance related faults using ECP-6. We can potentially use the storage for one of the ECP entries for storing Berger code and still have space left for implementing ECP-5. Our analysis will assume a DIMM structure similar to SECDED DIMM, whereby each access for a line of 64 byte also brings in additional meta-data of 8 bytes.[5]

C. Error Correction with Detection and Retry

If an error is detected, we simply rely on a fallback mechanism of reading with a higher latency (normal latency) to correct errors, as shown in Figure 8. Thus, the PCM memory system must have support for two read latencies: First, a shortened latency as afforded by Early Read and Second, the normal latency. If the first try with shorter latency fails, then we try the read request again with normal latency. This allows Early Read to provide low latency in common case, and still provides correct results in the uncommon case of data error.

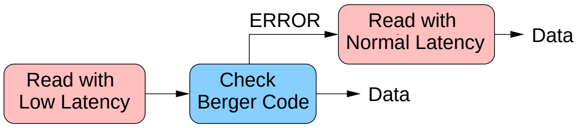

Fig. 8. Error correction by retrying reading data with a longer latency.

[4] While Berger Codes guarantee detection of all unidirectional errors, they can also always detect single bidirectional errors. To detect multiple bidirectional errors from faulty cells being overlapped with unidirectional sensing errors, one may use error detection codes such as CRC [28, 29].

[5] The logic for implementing Berger code incurs negligible area and latency. Given a bus size of 72 bits, we count the number of 1s in each bus transfer (72 bits) and accumulate it for all the transfers for a given line. Counting the number of 1s in a 72-bit word incurs an area overhead of few hundred logic gates and a delay of less than 25FO4 (less than 2 cycles on a 3GHz processor). We include this latency in our calculations.

D. Model for Selecting the Target Read Latency

We can tailor the sensing time of Early Read by tuning the read circuitry for appropriate sense resistance. Lower sense resistance would give lower sensing time but higher probability of retry. Whereas, higher sense resistance would give comparatively higher sensing but at a reduced probability of retry. We want to select an operating point for Early Read that balances both these effects, to give an overall reduction in read latency. We develop a model to converge on the best operating point for Early Read.

Let T_{Read} be the total reading time for the baseline. Let T_{Read_ER} be the reading time for Early Read. Let *r* is the probability that a read request may flag unidirectional errors and result in a retry. The effective read latency (T_{Read_ER-Eff}) can be computed as shown in Equation 2.

$$T_{Read\_ER-Eff} = (1-r)\cdot T_{Read\_ER} + r\cdot(T_{Read\_ER} + T_{Read}) \quad (2)$$

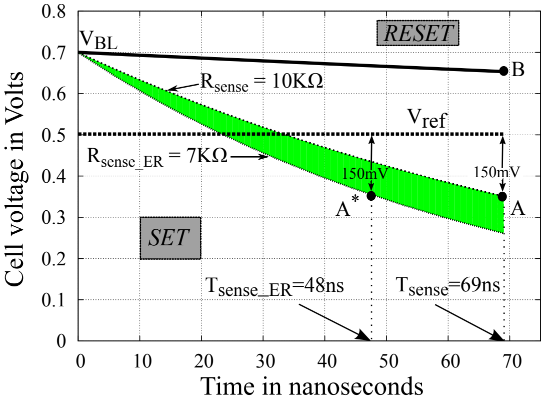

Fig. 9. Targeting a lower SET resistance (from 10KΩ to 7KΩ) reduces sensing time from 69ns to 48ns (while maintaining the same voltage margin of 150mV from either curves).

1) Effects on Sensing Time: We first discuss how the point of operation affects the sensing time of early read. Figure 9 shows the cell voltage for the baseline tuned to a SET resistance of 10KΩ and RESET resistance of 100KΩ. The sensing time is the time at which there is sufficient difference in voltage for the two states. We keep a margin of 150mV to account for variation in reference voltage in either direction. So, the baseline system will have a T_{sense} of 69ns.

Now, instead of the SET cell of 10KΩ, if we use a SET cell of 7KΩ then the cell discharges faster, as shown by the line corresponding to A*. Now, for the same voltage difference of 300mv we will be expected to wait only for 48ns. Similar analysis can be performed for other points of operation for the selected value of the target SET resistance.

2) Effects on Retry Probability: The overall read latency must be balanced for a point where the gains from reducing the sensing time does not get exhausted by the higher latency incurred due to retry. Table III shows the probability of retry and the effective read latency per Equation 2 We see that Rsense equals 7KΩ, we get a retry probability of 0.5%, and the lowest overall read latency. Hence, we use 7KΩ as the default value in our evaluations for Early Read.

TABLE III. EFFECTIVE READ LATENCY FOR DIFFERENT R_{sense}.

R_{sense}	T_{sense}	P(retry)	Effective T_{rd} (incl T_{PRE}, T_{BUS})
10KΩ	69 ns	0.0%	80 ns
8KΩ	55 ns	0.0%	66 ns
7.5KΩ	51 ns	0.0%	62 ns
7KΩ	**48 ns**	**0.5%**	**60 ns**
6.5KΩ	44 ns	40%	90 ns

E. Impact of Early Read on Execution Time

Early Read reduces the read latency of PCM from 80ns to 60ns. The 25% reduction in read latency also translates to performance. Figure 10 shows the speedup from Early Read. The bar labeled *Gmean* indicates the geometric speedup for all the workloads. Workloads such as mcf_r spends more than 80% of their execution time in memory, hence they obtain a significant speedup (20% lower execution time with 25% faster memory). Overall, we observe that Early Read is quite effective, and provides an average speedup of 17%.

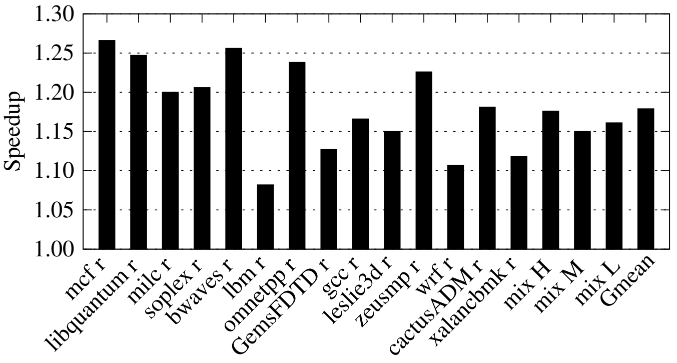

Fig. 10. Speedup with Early Read. On average, it provides 17% speedup.

V. TURBO READ

Early Read targets the sensing time directly in order to improve the latency of read operations. However, Early Read relies on the memory system to support two types of read latency (low latency and normal latency). Our second proposal, *Turbo Read*, targets the sensing voltage as a means to reduce the sensing time, without requiring a dual latency support from the memory system. We first describe the constraints that limit the sensing voltage, then how increasing the sensing voltage can reduce sensing time at the expense of causing data errors. We then provide means of tolerating data errors that happen because of Turbo Read.

A. Read Disturb and Raising the Read Voltage

Write operations to a PCM cell are performed by electrical heating using current pulses. The sensing voltage for a read operation (V_{rd}) is kept at a level such that the probability of accidentally writing to the cell while reading is negligibly low. An accidental change in the state of the cell while doing a read operation is called as *read disturb*. If the cell is in the SET state, the worst-case read current flowing through any cell is much lesser than the current necessary to create any appreciable heating in the cell, and thus there is virtually no chance that the cell will transition to the RESET state, even at higher read voltages. On the other hand, if the cell is in the RESET state, the worst-case peak current flowing through any cell may be sufficient to cause switching to SET.

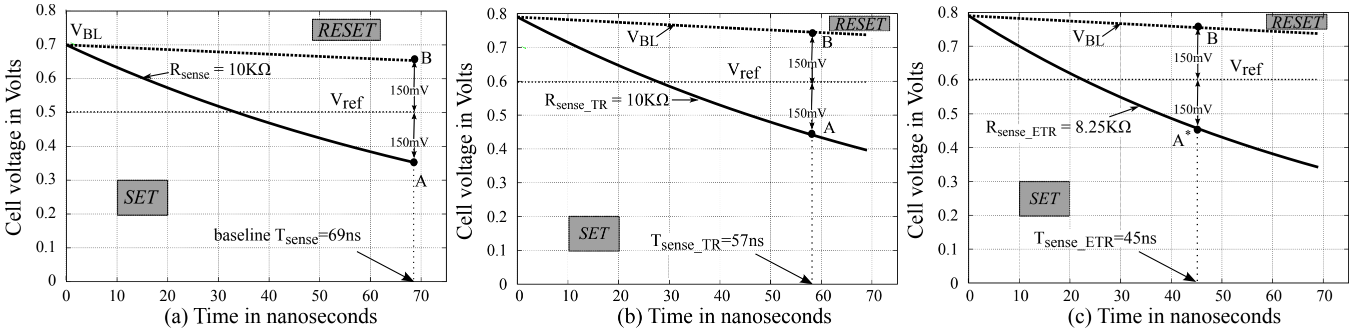

Fig. 11. Determining the sensing time of a system for (a) Baseline (T_{sense}) (b) Turbo Read (T_{sense_TR}) (c) Turbo Read + Early Read (T_{sense_ETR}).

The insight in Turbo Read is to keep a sensing voltage higher than normal; to reduce the sensing time, but low enough that the read disturb errors can be mitigated in a practical and cost-effective manner. To understand the relationship between the sensing voltage and error rate due to read disturb, we follow the study that characterizes this data for PCM devices [18]. Their study shows that an increase in sensing voltage by 30mV increases the likelihood of read disturb errors by 3 orders of magnitude. We assume that for the baseline system the bit error rate due to read disturb is 10^{-18} (so that the likelihood of a flipped bit on each read of a cache line can be in the range of 10^{-16}). For our studies, we increase the sensing voltage from 0.70V to 0.79V[6], which would increase the read disturb probability for a bit from 10^{-18} to 10^{-9}.

B. Handling Read Disturb with ECC Code

Read disturb faults occur while performing a read operation at a higher than specified sensing voltage. However, these errors may not be visible immediately in the data read from the PCM array. The corrupted bit may stay in the line, and may become visible only on subsequent read to that line.

To mitigate the errors that happen because of read disturb, we can equip the memory with error correction codes (ECC). However, we want to keep the overhead of ECC to a manageable level, and avoid using high strength ECC codes as they incur significant cost and complexity. A memory system that is provisioned to correct K errors per line will give an uncorrectable error when the line has more than K errors. To converge on a suitable value of K we study the probability that the line has an uncorrectable error for different value of Bit Error Rate (BER), as shown in Table IV.

TABLE IV. PROBABILITY THAT LINE HAS K ERRORS AT GIVEN BER

BER Read disturb	Prob (Line has K errors)			
	k=1	k=2	k=3	k=4
10^{-8}	5.1×10^{-6}	1.3×10^{-11}	2.2×10^{-17}	2.9×10^{-23}
10^{-9}	5.1×10^{-7}	1.3×10^{-13}	**2.2×10^{-20}**	2.9×10^{-27}
10^{-10}	5.1×10^{-8}	1.3×10^{-15}	2.2×10^{-23}	2.9×10^{-31}

For a BER of 10^{-9}, we observe that the probability that the line will have 3 or more errors is 10^{-19}, which is well below the target range we seek (10^{-16}). Therefore, we conclude that simply using ECC code that can correct two errors per line (ECC-2) would be sufficient for Turbo Read. The storage required for ECC-2 would be 20 bits per line.

Fortunately, prior work has looked at efficiently integrating ECC codes in PCM memories. The FREE-p design [30] used ECC-2 and ECP-4 in order to tolerate both soft errors and hard errors in PCM memories. While they did not look into redesigning the PCM system given the error correction capability, in our paper we leverage the existence of ECC to design a better-than-worst-case PCM system. For the studies with Turbo Read, we will assume a FREE-p like architecture that supports ECC-2 with every PCM line.[7]

C. Turbo Read Operation and Synergy with Early Read

To implement Turbo Read, the sensing circuit is tuned to operate with higher V_{BL} and V_{ref} for all read operations. When the line is read, an error may occur that flips the bit being read. Such an error may not be visible during the ongoing read operation. However, on a subsequent read operation for the same line, the ECC code with the line will detect the error, correct it, and rewrite the corrected value to memory (to avoid accumulation of errors), and also provide this corrected value as the output of the read operation. As long as the line does not encounter more than two errors, ECC-2 will correct it.

Turbo Read and Early Read are synergistic and can be combined. Unfortunately, errors can happen in both directions in this system, Fortunately, the ECC-2 associated with the line can be used for correcting errors that happen from both Early Read as well as Turbo Read. The restriction however, is that the BER for the two types of errors combined should not be significantly higher than 10^{-9}. For example, if Turbo Read has a BER of 10^{-9} and we select the sensing resistance for Early Read such that the BER for Early Read is also 10^{-9}, then the combined error rate for the system will be 2×10^{-9}. From Table IV, we can estimate that for a BER of 2×10^{-9}, the likelihood that the line will have 3 or more errors will be approximately 10^{-18}, which is still well below the target error rate we seek (10^{-16} for the line). To obtain a BER of 10^{-9} with Early Read, the sensing resistance for Early Read must be kept at 8.25KΩ (instead of the 7KΩ used in Section IV). For the configuration that combines Turbo Read and Early Read, we will use a sensing resistance of 8.25KΩ.

D. Analysis for Sensing Time

We perform analysis of the sensing time to determine the read latency of a system with Turbo Read (with and without Early Read). Figure 11 shows the cell voltage for the baseline tuned to a SET resistance of 10KΩ and RESET resistance of 100KΩ and a V_{BL} of 0.7V. The sensing time is the time

[6] Read current is only 40μA compared to write current of 300μA and write current impacts PCM lifetime [3]. Fortunately, Turbo Read causes only a nominal increase in read current (45μA), negligibly impacting lifetime

[7] For BER of 10^{-9}, less than 1 read disturb error happens per 1 million reads. Such writes have a negligible impact on endurance ($\ll$ 0.01%)

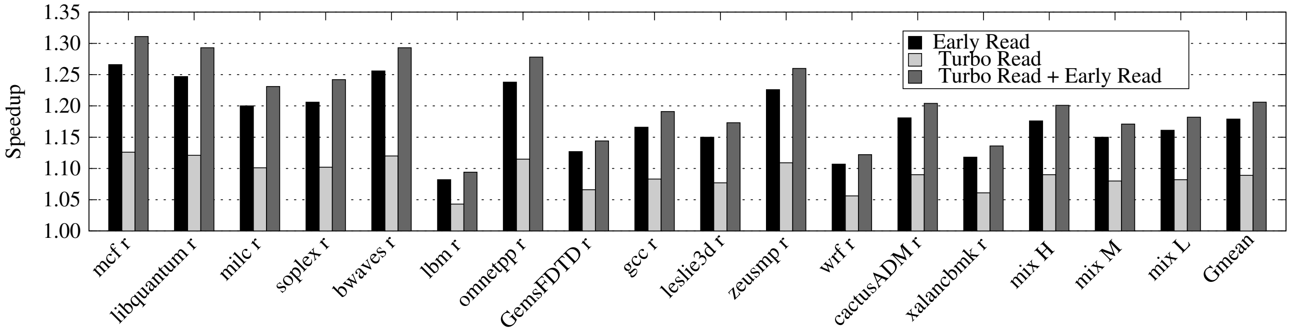

Fig. 12. Speedup with Early Read alone, Turbo Read alone, and combination of Early Read + Turbo Read. Early Read alone provides an average speedup of 17% and Turbo Read alone provides an average speedup of 9.7%. The combination of Turbo Read + Early Read provides an average speedup of 21%.

at which there is sufficient difference in voltage for the two states. We keep a margin of 150mV from either direction to account for variation in reference voltage. The time at which the expected difference from both the curves is 150mV is chosen as the time to latch the data value. Thus, the baseline has a T_{sense} of 69ns. The overall read latency for the baseline, including T_{PRE} and T_{BUS} will be 80ns.

For the system with Turbo Read, we increase the V$_{BL}$ to 0.79V and again try to find the point where the expected value of voltage decaying from both the SET (10KΩ) and RESET (100KΩ) are separated from the reference voltage by 150mV. We revise the reference voltage (V$_{ref}$) from 0.502V for the baseline to 0.596 for Turbo Read. As shown in Figure 11(b), Turbo Read provides a T_{sense} of 57ns. The, overall read latency for one read operation with Turbo Read, including T_{PRE} and T_{BUS} will be 68ns, 15% lower than the baseline.

To combine Turbo Read and Early Read, we change the V$_{BL}$ to 0.79V and instead of using the SET cell of 10KΩ, we use a SET cell of 8.25KΩ. A lower resistance cell allows the voltage to decay faster, as shown by the line corresponding to A* in Figure 11(c). For maintaining the voltage margin of 150mv we will be now need only 45ns[8]. The read latency for one read operation for a system that combines both Turbo Read with Early Read, including T_{PRE} and T_{BUS} will be 56ns, 30% lower than the baseline.

E. Impact on Execution Time

Figure 12 shows the speedup from Early Read, Turbo Read, and a combination of Early Read and Turbo Read. Note that our proposals reduce the isolated memory latency, independent of the workloads. For workloads that spend a significant portion of execution time in memory, the reduction in memory latency results in significant speedups. For example, mcf_r spends more than 80% of the execution time in memory, which causes it to get 1.32x speedup (25% reduction in execution time) due to 30% reduction in memory latency with Early Read + Turbo Read. On average, Early Read improves performance by 17%, Turbo Read improves performance by 9.7%, and the combination improves performance by 21%.

F. Sensitivity to Target Bit Error Rate (BER)

We perform our studies assuming a target BER rate of 10^{-16}. In this section, we vary the BER to evaluate the effectiveness of our schemes across different BER. Figure 13 shows the speedup for Early Read, Turbo Read and their combination as the target BER of the system is varied from 10^{-14} to 10^{-20}. Speedup due to Early Read varies from 14% at BER of 10^{-14} to 26% at BER of 10^{-20}. This is because at low BER such as 10^{-20} the value of R$_{ref}$ is higher which increases the sensing time. Fortunately, the target resistance for Early Read (BER of 10^{-5}) is not dependent on R$_{ref}$, enabling an even higher speedup. The combination Turbo Read and Early Read shows speedup from 16% BER of 10^{-14} to 29% at BER of 10^{-20}. Turbo Read consistently shows a speedup of nearly 10% as reference voltage and target resistance vary with BER. Thus, our schemes are even more effective and robust for systems that are designed to target much lower BER.

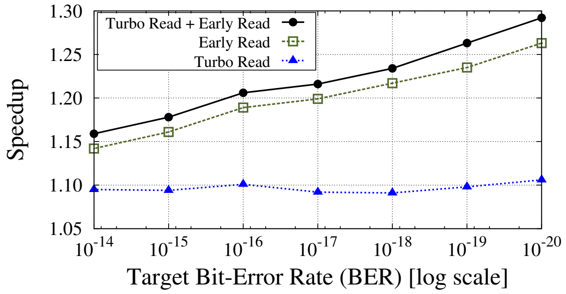

Fig. 13. Speedup for different Target BER. Our schemes are even more effective for systems targeting lower RBER.

G. Impact of Tolerating Drift

The resistances of PCM cells increase over time due to drift. We use values from [31] which show that rate of increase dependent on the resistance of the cell, in that lower resistance values drift less than higher resistance values. So, drift has a smaller impact on cells around the resistance of Early Read than for the reference resistance of the system. To mitigate drift, the system will need to be designed to have an even higher R_{ref} than what we have assumed and our solutions will become even more effective as they can exploit these margins too. Figure 14 shows the speedup for a system that has been provisioned to handle a margin of 1 day, 100 days, and 5 years of drift. Turbo Read does not show benefits as reference voltage and target resistance vary more than the baseline. The performance of Early Read + Turbo Read increases from 21% (without drift) to 27% (for a system designed to handle drift of 5 years). Thus, our combined proposal becomes even more effective when the margins for drift are considered.

[8] In our designs, we conservatively maintain the same voltage margins from the reference voltage as the baseline. However, given that the system is provisioned with ECC-2 per line, a modest variation in reference voltage can be tolerated. As it will result in a small chance of error, which can be corrected by the ECC-2. We show that even without such optimizations, our proposed designs can reduce read latency significantly. More gains are possible for further optimizing for reference voltage variation as well.

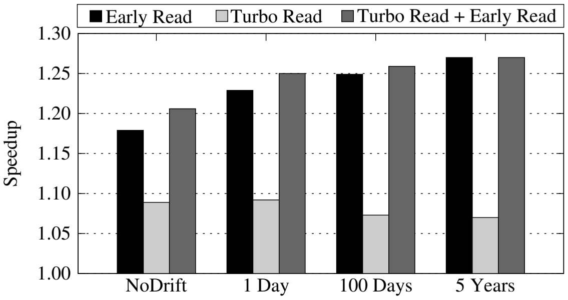

Fig. 14. Impact of drift tolerance. Turbo Read + Early Read becomes even more effective by expoliting variability in drift.

VI. Avoiding Latent Errors in Turbo Reads

Thus far, for the reliability analysis of Turbo Read we have assumed that once a line suffers an error due to read disturb, a second read request for the line will be able to detect the error and correct for it (the likelihood of a single request causing three or more read disturb errors in a line is negligibly small). This would be a safe assumption if for every read operation the memory system sensed only the requested line and no other lines. However, memory system is organized at a row granularity, with PCM memories typically having a row buffer of 256-512 bytes (4 to 8 cache lines) [3, 22]. When request for one line is serviced (say line X) then all the neighboring lines of X belonging to the same row buffer also get sensed. However, only the requested line is sent to the memory controller, and undergoes ECC correction. All the other unreferenced line in the row buffer may continue to accumulate errors due to read disturb, and if these lines remain unreferenced for a long time the number of errors could exceed the error correction capability of the line. A later reference to this line would result in data error. While such pathological access patterns are not typical in general applications, they can happen in malicious or badly written code. We discuss the patterns that can cause latent errors with Turbo Read and a low-cost scheme to protect the memory against such errors.

```
for(i=0; i<N; i++){
   access line X
   clflush(X)
   access line Y   (Same bank,diff row)
   clflush(Y)
}
access all lines in Row of X and Y
```

Fig. 15. Code for causing latent errors in a system employing Turbo Read.

A. Code for Latent Errors within Row

For an access pattern to cause a latent error with Turbo Read, there are two requirements. First, there is at least one line that remains referenced for a long time in a frequently accessed row buffer. Second, that unreferenced line gets used eventually. Figure 15 shows a potential code segment that can obtain this behavior (similar to the code in [32]). We assume that lines X and Y are located in different rows of the same memory bank. Request for line X and Y, each activate their row buffers for exactly one line. This process is repeated a large number of times N. Subsequently, all the lines of the row buffer are accessed to read the (potentially) erroneous line.

Figure 16 shows the impact of this kernel on Turbo-Read at a BER of 10^{-9}. The line labeled *NoPRS* represents the baseline with Turbo Read. At 10 million iterations of the attack kernel, the likelihood that the silent line would get an uncorrectable error is close to 100%. For the attack kernel, each iteration requires approximately 100ns (two reads, one each for X and Y). So the time to cause an error is nearly 1 second.

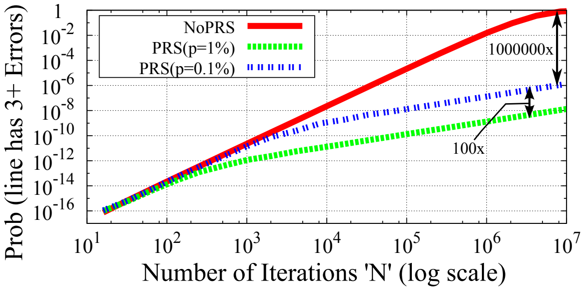

Fig. 16. Probability that a line develops 3+ errors due to the attack kernel. PRS reduces the likelihood of failure under attack by 10^6x to 10^8x.

B. Probabilistic Row Scrubbing

The vulnerability from latent errors can be avoided by periodically scrubbing the memory. During a scrub each line is read and if line has errors it undergoes an ECC based correction and written back to memory. However, to avoid failure with our attack kernel, the period for a memory scrub should be under 1 second, which is prohibitively expensive. Instead of doing scrub for the entire memory, we propose an efficient scheme that applies the mitigation only to the rows that get accessed frequently. We call this scheme *Probabilistic Row Scrubbing (PRS)*. PRS is inspired from our prior work on Row Hammering [33]. With PRS, on every read access, the entire row gets scrubbed with a small probability (p). For resilience, we want p to be high, but to avoid the performance overhead associated with scrubbing, we want p to be low. We analyze $p = 1\%$ and $p = 0.1\%$, so that on average a row scrub will happen once every 100 (or thousand) read requests. A scrub operation transfers four to eights lines (depending on row buffer size) from memory instead of one line for regular read operation. PRS with such a low probability of scrub has negligible impact on performance (on average, less than 0.5% for $p = 1\%$, and is already incorporated in Figure 12).

C. Effectiveness of Probabilistic Row Scrubbing

Figure 16 shows the probability that the line will have three or more errors due to read disturb as number of iterations (N) increase, with PRS enabled ($p = 1\%$ and $p = 0.1\%$). At N=10M, for the baseline there is a 99% chance that the line has 3+ errors. PRS (with p=0.1%) reduces this by six orders of magnitude to approximately 10^{-6}, and with p=1% this reduces by eight orders of magnitude to approximately 10^{-8}. The expected time for the line to fail under PRS would be 10 days with $p = 0.1\%$ and 3 years with $p = 1\%$. Thus, PRS is effective at making the memory tolerant to latent errors.

It must be noted though, that in reality, the episode of requesting the same line continuously from memory does not happen with typical applications because of the presence of caches (that contain space of storing tens of thousands of cache lines). So, we expect that, on average, there would be at least a few hundreds (if not thousands) of request between frequent accesses to the same row. Therefore, in practice, even badly written access patterns will be slowed down by 100x-1000x, resulting in much higher time to failure under attack.

VII. Impact on Energy and Power

We analyze the effectiveness of our proposed designs on the memory system power and energy. For Early Read, the read operations finish earlier so the read power is consumed for a shorter period of time. For, Turbo Read increases the sensing voltage (from 0.7V to 0.79V) so each read request consumes higher power. Furthermore, systems with Turbo Read also incur small amount of extra power for PRS. Figure 17 shows the speedup, energy in memory system (MemEnergy), power in memory system (MemPower), and the Energy Delay Product (EDP) of the system for our proposed designed, all normalized to the baseline. For EDP calculations, we assume that the memory system accounts for 25% of the total power of the baseline, and the processor and L4 cache consumes 75%.

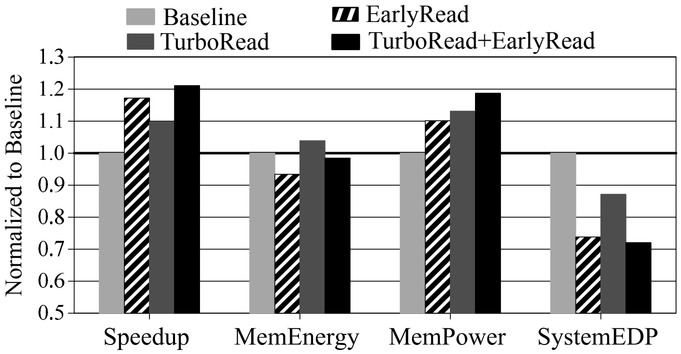

Fig. 17. Speedup, Memory Energy and Memory Power, and System wide EDP for our proposed scheme.

Early Read not only provides significant speedup, it is also energy efficient. As our schemes reduce the execution time significantly, memory power is increased (as similar amount of activity is now done in shorter time). Overall, Early Read reduces the system EDP by 24.5%, Turbo Read by 14%, and the combination of Turbo Read and Early Read by 28.5%.

VIII. Related Work

To our knowledge this is the first work that exploits PCM cell variability to improve read latency using architectural techniques. However, there are a few studies that have looked at exploiting variability in write characteristics to improve write performance and endurance. Zhang et al [34] studied the process variation in PCM memories and showed that over-programming causes significant power overheads and endurance degradation. They proposed to tune the programming (write) current to adapt to process variation. However, they did not study any scheme to improve the read latency of PCM memories. Similar approach to optimizing write performance by exploiting variability has been studied for Flash technology [35]. Similarly, Childers et. al [36] investigated techniques to improve the write performance of MLC cell, by exploiting the fact that not all cells require iterative programming (or require worst-case number of write iterations). Instead of using worst case number of write iterations for all cells, they rely on ECC to correct bits that incur long-latency to write. However, their scheme does not improve latency of read operations, and neither it is applicable to SLC memories.

Single Level Cell (SLC) incurs lower latency than MLC. Using this insight, Morphable Memory System (MMS) proposed by Qureshi et. al selectively morphs MLC into SLC depending on workload capacity and reduces the overall latency of PCM [37]. However, MMS does not optimize the SLC latency. In contrast, our proposal can optimize the read latency of SLC memories as well.

Recent ideas on optimizing DRAM such as ‘Tiered-Latency DRAM’ lower DRAM latency by reducing bit-line capacitance using shorter bit lines dynamically [38]. We can apply bit-line tiering to PCM to reduce bit-line capacitance as well. Since we target the variation in cell resistance, the two ideas can be combined to get an even lower read latency.

Several studies exploit variability in retention time of DRAM cells to avoid doing refresh operations [39, 40, 41, 42]. We can potentially use such schemes to tolerate cells that cause failure in Early Read and Turbo Read, assuming that the characteristics of the such cells do not change over time.

IX. Summary

Phase Change Memory (PCM) is one of the leading candidates in emerging memory technologies. Unfortunately, PCM has higher latency than DRAM, which degrades system performance. In this paper, we studied architectural techniques to improve the read latency of PCM by exploiting the inherent variation in resistance characteristics of PCM cells. Rather than conventional designs that try to hide variability, we contend that PCM systems should be designed for embracing variability and designing for better-than-worst-case scenarios. To that end, we proposed two solutions:

1) *Early Read*, a design that advocates reducing sensing latency by latching the output of the sense amplifiers earlier than the specified time period. While this is effective for most cases, infrequently this may cause errors. Fortunately, such errors are unidirectional, and we propose a combination of Berger Code and retry mechanism to efficiently mitigate errors. Early Read reduces the read time by 25% and provides an average speedup of 17%.
2) *Turbo Read*, a design that advocates using higher sensing voltage as a mechanism to reduce the read sensing time. We propose mitigation techniques (ECC and Probabilistic Row Scrubbing) for the read disturb errors that happen due to Turbo Read. We show that Turbo Read and Early Read can be combined. The combined scheme reduces the read latency by 30% and provides an average speedup of 21%.

We show that not only are these schemes effective at reducing read latency, and improving system performance, they also provide significant improvement in system EDP (a 28% reduction). The proposed schemes can be implemented with very little changes to the memory system. While we conduct our studies for SLC PCM system, the proposed ideas can also be applied to MLC systems and other memory technologies.

Acknowledgments

We thank Hadi Esmailzadeh for his comments on the initial version of this work. We also thank Aseem Grover, Vinson Young, and David Roberts for their feedback. This work was supported in part by the Center for Future Architectures Research (C-FAR), one of the six SRC STARnet Centers, sponsored by MARCO and DARPA, a gift from VMware, and the Department of Science and Technology, India.

References

[1] M. K. Qureshi, S. Gurumurthi, and B. Rajendran, ser. Synthesis Lectures on Computer Architecture. Morgan and Claypool Publishers, 2011.

[2] M. K. Qureshi, V. Srinivasan, and J. A. Rivers, "Scalable high performance main memory system using phase-change memory technology," in *ISCA-2009*.

[3] B. C. Lee, E. Ipek, O. Mutlu, and D. Burger, "Architecting phase change memory as a scalable dram alternative," in *ISCA-2009*.

[4] M. K. Qureshi, J. Karidis, M. Franceschini, V. Srinivasan, L. Lastras, and B. Abali, "Enhancing lifetime and security of pcm-based main memory with start-gap wear leveling," in *MICRO-2009*.

[5] N. H. Seong, D. H. Woo, and H.-H. S. Lee, "Security refresh: Prevent malicious wear-out and increase durability for phase-change memory with dynamically randomized address mapping," in *ISCA-2010*.

[6] S. Schechter, G. H. Loh, K. Strauss, and D. Burger, "Use ECP, not ECC, for hard failures in resistive memories," in *ISCA-2010*.

[7] N. H. Seong, D. H. Woo, V. Srinivasan, J. Rivers, and H.-H. Lee, "Safer: Stuck-at-fault error recovery for memories," in *MICRO-2010*.

[8] M. K. Qureshi, "Pay-as-you-go: Low-overhead hard-error correction for phase change memories," in *MICRO-2011*.

[9] M. Qureshi, M. Franceschini, and L. Lastras-Montano, "Improving read performance of phase change memories via write cancellation and write pausing," in *HPCA-2010*.

[10] M. Qureshi, M. Franceschini, A. Jagmohan, and L. Lastras, "Preset: Improving performance of phase change memories by exploiting asymmetry in write times," in *ISCA-2012*.

[11] A. Hay, K. Strauss, T. Sherwood, G. H. Loh, and D. Burger, "Preventing pcm banks from seizing too much power," in *MICRO-2011*.

[12] L. Jiang, Y. Zhang, B. Childers, and J. Yang, "Fpb: Fine-grained power budgeting to improve write throughput of multi-level cell phase change memory," in *MICRO-2012*.

[13] H. Li and Y. Chen, *Nonvolatile Memory Design: Magnetic, Resistive, and Phase Change*. Taylor & Francis, 2011.

[14] S. Hanzawa *et al.*, "A 512kb embedded phase change memory with 416kb/s write throughput at 100μA cell write current," in *Solid-State Circuits Conference, 2007. ISSCC 2007. Digest of Technical Papers. IEEE International*, Feb.

[15] B. Wicht, T. Nirschl, and D. Schmitt-Landsiedel, "Yield and speed optimization of a latch-type voltage sense amplifier," *Solid-State Circuits, IEEE Journal of*, vol. 39, no. 7, pp. 1148–1158, July 2004.

[16] *Understanding The New Bit Error Rate DRAM Timing Specifications*, JEDEC: Agilent Technologies, Server Memory Forum 2011.

[17] J. Berger, "A note on error detection codes for asymmetric channels," *Information and Control*, vol. 4, no. 1, pp. 68 – 73, 1961.

[18] S. Lavizzari, D. Ielmini, D. Sharma, and A. Lacaita, "Transient effects of delay, switching and recovery in phase change memory (pcm) devices," in *Electron Devices Meeting, 2008. IEDM 2008. IEEE International*.

[19] S. Lavizzari, D. Sharma, and D. Ielmini, "Threshold-switching delay controlled by 1/f current fluctuations in phase-change memory devices," *Electron Devices, IEEE Transactions on*, vol. 57, no. 5, pp. 1047–1054, May 2010.

[20] A. Faraclas, N. Williams, A. Gokirmak, and H. Silva, "Modeling of set and reset operations of phase-change memory cells," *Electron Device Letters, IEEE*, vol. 32, no. 12, pp. 1737–1739, Dec 2011.

[21] J. Yue and Y. Zhu, "Exploiting subarrays inside a bank to improve phase change memory performance," in *Design, Automation Test in Europe Conference Exhibition (DATE)-2013*.

[22] Y. Choi *et al.*, "A 20nm 1.8v 8gb pram with 40mb/s program bandwidth," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2012 IEEE International*.

[23] G. Close *et al.*, "A 256-mcell phase-change memory chip operating at 2+ bit/cell," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 60, no. 6, pp. 1521–1533, June 2013.

[24] G. De Sandre *et al.*, "A 4 mb lv mos-selected embedded phase change memory in 90 nm standard cmos technology," *Solid-State Circuits, IEEE Journal of*, vol. 46, no. 1, pp. 52–63, Jan 2011.

[25] D. Mantegazza, D. Ielmini, A. Pirovano, A. Lacaita, E. Varesi, F. Pellizzer, and R. Bez, "Explanation of programming distributions in phase-change memory arrays based on crystallization time statistics," *Solid-State Electronics*, vol. 52, no. 4, pp. 584 – 590, 2008, special Issue: Papers Selected from ULIS 2007 - Papers Selected from the ICMTD 2007.

[26] T. Austin and V. Bertacco, "Deployment of better than worst-case design: solutions and needs," in *Computer Design: VLSI in Computers and Processors, 2005. ICCD 2005. Proceedings. 2005 IEEE International Conference on*, pp. 550–555.

[27] G. M. Koob and C. Lau, *Foundations of Dependable Computing: System Implementation*.

[28] P. J. Nair, D. A. Roberts, and M. K. Qureshi, "Citadel: Efficiently protecting stacked memory from large granularity failures," in *MICRO-2014*.

[29] J. Sim, G. H. Loh, V. Sridharan, and M. O'Connor, "Resilient die-stacked dram caches," in *ISCA-2013*.

[30] D. H. Yoon, N. Muralimanohar, J. Chang, P. Ranganathan, N. Jouppi, and M. Erez, "Free-p: Protecting non-volatile memory against both hard and soft errors," in *HPCA-2011*.

[31] M. Awasthi, M. Shevgoor, K. Sudan, B. Rajendran, R. Balasubramonian, and V. Srinivasan, "Efficient scrub mechanisms for error-prone emerging memories," in *HPCA-2012*.

[32] 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 *ISCA-2014*.

[33] D. Kim, P. Nair, and M. Qureshi, "Architectural support for mitigating row hammering in dram memories," in *Computer Architecture Letters*, 2015.

[34] W. Zhang and T. Li, "Characterizing and mitigating the impact of process variations on phase change based memory systems," in *MICRO-2009*.

[35] C. Race, Y. P. Kim, and R. Bowman, "Controlling program parameters to increase nand flash life for ssd applications," in *NVMW*, 2014.

[36] L. Jiang, B. Zhao, Y. Zhang, J. Yang, and B. Childers, "Improving write operations in mlc phase change memory," in *HPCA-2012*.

[37] M. K. Qureshi, M. M. Franceschini, L. A. Lastras-Montaño, and J. P. Karidis, "Morphable memory system: A robust architecture for exploiting multi-level phase change memories," in *ISCA-2010*.

[38] D. Lee, Y. Kim, V. Seshadri, J. Liu, L. Subramanian, and O. Mutlu, "Tiered-latency dram: A low latency and low cost dram architecture," in *HPCA-2013*.

[39] R. Venkatesan, S. Herr, and E. Rotenberg, "Retention-aware placement in dram (rapid): software methods for quasi-non-volatile dram," in *HPCA-2006*.

[40] J. Kim and M. Papaefthymiou, "Block-based multi-period refresh for energy efficient dynamic memory," in *ASIC/SOC Conference, 2001. Proceedings. 14th Annual IEEE International*, 2001.

[41] J. Liu, B. Jaiyen, R. Veras, and O. Mutlu, "Raidr: Retention-aware intelligent dram refresh," in *ISCA-2012*.

[42] P. J. Nair, D.-H. Kim, and M. K. Qureshi, "Archshield: Architectural framework for assisting dram scaling by tolerating high error rates," in *ISCA-2013*.