Captopril: Reducing the Pressure of Bit Flips on Hot Locations in Non-Volatile Main Memories

Majid Jalili* and Hamid Sarbazi-Azad*†
*HPCAN Lab, Department of Computer Engineering, Sharif University of Technology, Tehran, Iran
†School of Computer Science, Institute for Research in Fundamental Science (IPM), Tehran, Iran
majalili@ce.sharif.edu, azad@{ipm.ir, sharif.edu}

***Abstract*—High static power consumption and insufficient scalability of the commonly used DRAM main memory technology appear to be tough challenges in upcoming years. Hence, adopting new technologies, i.e. non-volatile memories (NVMs), is a proper choice. NVMs tolerate a low number of write operations while having good scalability and low static power consumption. Due to the non-destructive nature of a read operation and the long latency of a write operation in NVMs, designers use read-before-write (RBW) mechanism to mask the unchanged bits during write operation in order to reduce bit flips. Based on this observation that some specific locations of blocks are responsible for the majority of bit flips, we extend the RBW to further reduce the number of bit flips per write in the memory system. The results taken from full-system simulations reveal that our proposal, called Captopril, can reduce the number of bit flips by 21% and 9%, on average, compared to the baseline and state-of-the-art designs, respectively.**

I. Introduction

Due to the ever increasing demand for high capacity and fast memory system, using high density and low power non-volatile memory (NVM) is an inevitable choice. Among all proposed technologies, several NVMs including phase change memory (PCM), Spin-Torque Transfer RAM (STT-RAM) and Flash memory show appropriate potential for possible replacement of defacto DRAM and SRAM based memory systems. Although NVMs are very scalable and consume near-zero static power, they are slow and can tolerate relatively low number of write operations. Hence, short lifetime and long latency problems of these memories need to be addressed properly [3], [7].

A classic solution for extending the short lifetime of NVMs suggests masking the unchanged bits during the write operation through read-before-write (RBW) mechanism [7]. More clearly, when a write operation arrives to the memory controller, first, unchanged bits are determined by obtaining the current content of the block using an additional read operation. Then, the memory controller removes unchanged bits from write process to reduce bit flips. RBW is a practical solution because of two reasons: i) in contrast to DRAMs, a read operation in NVMs is not destructive, and ii) RBW has negligible cost in terms of performance since the latency of a write operation is 4-5 times longer than the latency of a read operation [7]. Additionally, RBW is an effective approach since typically in a memory system, a high amount of information redundancy is observed and therefore, a low number of bits are changed during write operation.

Several proposals are introduced in the literature to enhance the effectiveness of RBW [5], [7], [10], [17]. For instance, Flip-N-Write (FNW) partitions a block into some segments and by comparing the current content of the segment to the arrived content and its inverted form, selects the state that imposes lower bit flips. None of the proposed schemes, however, did pay attention to a phenomena in memory system called *bit flips non-uniformity* in this paper. More precisely, we observed that 42.4% of bit flips come from 128 out of 512 bit locations. This implies that there exists a considerable opportunity for bit flips reduction and thus energy saving in NVMs, if a proper solution to track the behavior of such hot locations exist.

To this end, we propose an architecture called *Captopril* to improve the energy efficiency of NVMs. We determine the hot locations of blocks by observing the behaviour of write operations in real applications. Then, to take advantage of such non-uniformity in bit flips, we present some rules to formulate hot locations. Knowing these locations, the memory controller can decide to write the content of hot locations in normal or inverted form. Our contributions can be summarized as follows:

- We investigate the non-uniformity of bit flips within a block in memory system and show that reducing bit flips in hot locations can considerably reduce NVM activities.
- We propose *Captopril* in order to benefit from this non-uniformity by determining the hot locations. *Captopril* uses some rules to determine hot locations. Our scheme partitions the block into some segments and each segment uses a rule that better covers hot locations. Knowing the hot locations, *Captopril* selects a writing scheme that reduces bit flips.
- We present an evaluation of the proposed technique using SPEC CPU 2006 and PARSEC workloads in gem5 for evaluating a PCM-based main memory. Our evaluations reveal that Captopril reduces the bit flips by 21% and 9%, on average, compared to the baseline and FNW schemes.

II. Background and Related Work

Phase Change Memory (PCM) is a resistive NVM that uses GST material to form memory cells. The resistant of GST

978-3-9815370-6-2/DATE16/

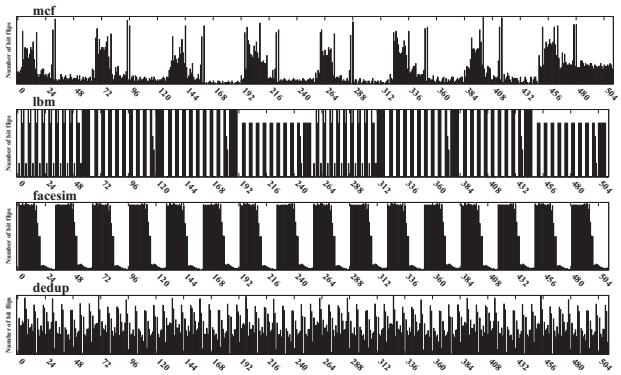

Fig. 1. Non-uniformity of bit flips in different applications: multi-program workloads (mcf and lbm) and multi-threaded applications (facesim and dedup).

ranges from KΩ to MegaΩ. Programming a PCM cell to the resistance values corresponding to ”0” and ”1” needs different pulses. To program a cell to ”0”, i.e reset state, a pulse with high amplitude and short duration is applied to the cell in order to increase its temperature above the melting point. After completing the formation process, the cell shows a high resistivity that can be interpreted as ”0” logic. To program a cell to ”1”, i.e. set state, a pulse with higher duration and lower amplitude (compared to the reset pulse) is used. The set pulse changes the formation of GST to a highly regular structure called crystalline and makes a very low resistance material, elucidated as logic ”1”. For reading the content of a cell, a weak pulse (a pulse that cannot change the state of GST but can measure the resistivity of cell) is used. By determining the cell resistance, with simple comparison to a touchstone resistance, the stored logic of the cell is determined [2].

CAFO [1] targets reducing the cost of writes by introducing a cost model for current write operation by considering the point that programming to 0 or 1 has different costs in terms of power and latency. Using compression for reducing the number of bit flips in non-volatile memories is proposed in [5], [13]. Due to the fact that read access energy in PCMs is content-dependent, authors in [6] proposed an efficient encoding technique for improving the energy efficiency of multi-level cell (MLC) RRAMs. FlipMin [14] uses COSET coding to enhance the lifetime of NVMs. Flip-Mirror-Rotate (FMR) [10] comprises of three components: adaptive Flip-N-Write (aFNW), Mirror-N-Write (MNW), and Rotate-N-Write (RNW). MNW and aFNW are word-level bit-write reduction schemes while RNW is an intra-word wear leveling scheme. FMR architecture cooperates with frequent pattern compression (FPC) scheme to simultaneously reduce bit-writes and wear level in NVMs. MFNW [9] is a Flip-N-Write encoding solution that reduces the average write energy and improves the endurance of MLC NVMs. WoM-SET [15] exploits WoM (write-once memory) code and reduces the number of RESETs per write and hence the write power.

III. Motivation

We observed considerable non-uniformity in bit flips during write operations that leave RBW and FNW unutilized within the memory system. Fig. 1 shows the number of bit flips in each bit position of a cache line for 4 applications: `mcf`, `lbm`, `facesim`, and `dedup`. As can be seen, some locations are very hot while some others are cold. Unfortunately, this hot locations which are in the minority, are the main barriers for system lifetime. Hence, to extend the lifetime, we should reduce the stress on theses location.

To estimate the effectiveness of a solution that cools down the hot locations, we removed the number of bit flips caused by top-rank locations from the total number of bit flips in Table.I. As can be seen, for `mcf`, the number of bit flips reduces by 92% when the 256 hottest locations are removed from total bit flips. On average, number of bit flips reduces by 6.93%, 12.93%, 18.73%, 23.79%, 42.48%, and 71.56% when the amount of flips for respectively 16, 32, 48, 64, 128, and 256 hottest locations are not considered. In other words, the contents of some locations which are in the minority, change with high probability. Hence, if we somehow reduce the bit flips in hot locations, it is expected that the lifetime and power consumption of the system improve significantly.

TABLE I
COMPARING THE PERCENTAGE OF BIT FLIPS REDUCTION WHEN IMPACTS OF HOTTEST LOCATIONS ARE NOT CONSIDERED.

Workloads	Percentage of bit flips reduction					
	16[a]	32	48	64	128	256
mcf	12.31	23.15	32.42	40.41	64.34	87.68
zeusmp	6.07	11.88	17.68	23.47	46.67	80.1
lbm	6.16	12.32	18.48	24.64	49.28	92.71
sjeng	6.92	13.57	19.97	26.09	49.16	87.82
leslie	5.41	10.46	15.06	19.26	35.71	63.28
milc	12.4	23.15	32.71	41.37	65.01	87.33
dedup	3.23	6.44	9.47	12.56	24.98	49.94
facesim	5.46	6.2	11.65	12.48	24.95	49.9
bodytrack	5.82	11.42	16.43	21.18	36.46	63.24
x264	4.4	8.81	11.38	14.11	26.2	50.98
fluidminate	8.05	14.89	20.78	26.14	44.56	74.21
Average	6.93	12.93	18.73	23.79	42.48	71.56

[a] Removing 16 hottest locations from each 512-bit block

IV. Captopril Mechanism

A. Abstract view

Minimizing the number of bit flips is a reasonable solution to tackle the endurance and power consumption problems of PCM. Knowing this fact that contents of specific locations change upon each write with high probability, it is not reasonable to update other cells every time. One bit can be used as an indicator to indicate the hot cells containing the original form of the data or its inverted form.

At the first glance, it seems that this idea is very similar to FNW. To clarify their difference, we provide an example to

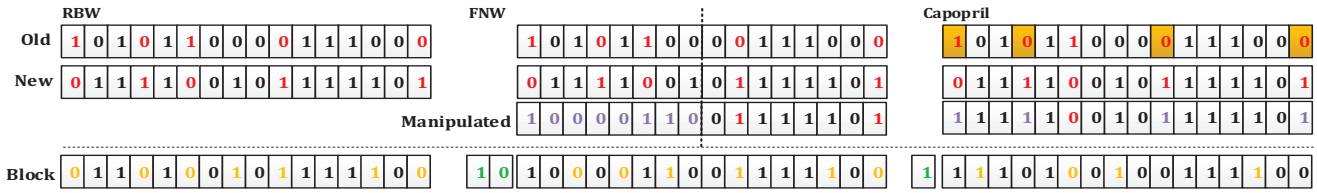

Fig. 2. Illustrating three mechanism: read-before-write (RBW), flip-n-write (FNW) and proposed techniques. (a) RBW reads the old data and by masking unmodified bits imposes 7 bit flips; (b) FNW partitions the block into 2 segments and detects the first partition that must be stored in the inverted form while the second partition uses the original format. It finishes the write with 6 bit flips; (c) Captopril, having the locations of hot cells, consiedrs the inverted form of new data for hot locations and results in 4 bit flips.

illustrate the underlying mechanism of *Captopril*. Fig.2 shows how RBW, FNW and Captopril strategies deal with a write. When RBW is employed, the controller reads the content of the block and then masks the unmodified bits. In this example, RBW changes 7 bits in the given 6-bit block. Using FNW, the block is partitioned into 2 segments. Then, by comparing the original form and inverted form of each segment with old content of the block, the one with less bit flips is selected for write operation. In this example, for first partition, inversion imposes lower number of flips while the second partition should be written in the original form. Altogether, adopting FNW, we have 6 bit flips in the block. Note that we need to keep track of each partition by one bit to determine whether it represents the original form or inverted form of data. However, when we adopt *Captopril*, if we know the position of hot bits (yellow boxes), we store the inverted form of the data simply for hot locations. Doing so, as can be seen in the figure, leads to 4 bit flips in the given block. One bit can distinguish the form of data (original or inverted) stored in the locations.

Our scheme needs to find the hot locations. But, if we want to track a sufficient number of hot locations by a mechanism involving pointers, the storage overhead is not tolerable. For example, if we want to track the 16 hottest locations in a 512-bit block, $16 \times 9 = 144$ bits storage overhead is imposed which is high. Hence, to solve this problem, we rely on an statistic-based solution. In other words, investigating the hot locations generated by different applications, we try to find some rules that can fomulate the hot locations.

Based on extensive experiment, we found that hot locations are found in following locations (bit locations are numbered linearly starting from the least significant bit of a block as 0, 1, 2, ..., blocksize - 1):

2X (or even locations): bit locations 0, 2, 4, 6, ...

Low Half Word (or LHW-n): partitioning a block into n words; hot bits sit in the least significant half of a word.

8X+4: bit locations 4, 12, 20, ...

We use the above three location patterns and ignore others to keep the storage overhead low. These 3 rules, in addition to the normal RBW, are considered in our scheme to reduce the number of bit flips through keeping their inverted forms. Table II shows the percentage of hot locations covered by above rules. On average, 69.18%, 21.73%, 47.02%, 52.98% and 50.28% of hot locations are of 2X, 8X+4 or LHW (LHW-8, LHW-16, LHW-32) types.

TABLE II
PERCENTAGE OF HOT BIT LOCATIONS IN 512-BIT BLOCKS.

Workloads	Percentage of hot locations				
	2X	8X+4	LHW-32	LHW-16	LHW-8
dedup	54.69	15.63	42.19	57.81	51.56
mcf	100	50	46.88	53.13	59.37
facesim	42.19	10.94	54.69	45.31	51.56
zeusmp	100	0	39.06	60.94	43.75
bodytack	51.56	12.5	40.63	59.38	35.94
lbm	100	48.44	53.13	46.88	40.62
sjeng	46.88	17.19	42.19	57.81	51.56
x264	54.69	10.94	39.06	60.94	56.25
fluidanimate	48.44	10.94	59.38	40.63	54.69
leslie	62.5	15.63	57.81	42.19	59.37
milc	100	46.88	42.19	57.81	48.44
Average	69.18	21.73	47.02	52.98	50.28

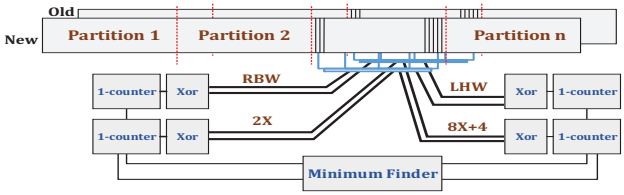

Fig. 3. Captopril mechanism: the block is partitioned into some parts (words) and then by considering the possible locations for hot bits and inverted form of corresponding cells, the number of bit flips using each rule is counted to find the minimum number of bit flips.

B. Architecture

To find the hot bit locations effectively based on the three mentioned rules, the block must be partitioned into some segments and for each segment the hot locations are found independently. In other words, as shown in Fig.3, firstly the block is partitioned into n segments and in each segment, we count the number of bit flips when different rules are employed. Then, we compare these numbers and among them the rule with lowest number of bit flips is selected for write. According to the number of partitions, the amount of storage overhead imposed to the system is different.

When a **write** is issued, firstly the content of corresponding block is obtained. Then, depending on which technique imposes lower number of bit flips, we select the proper storing

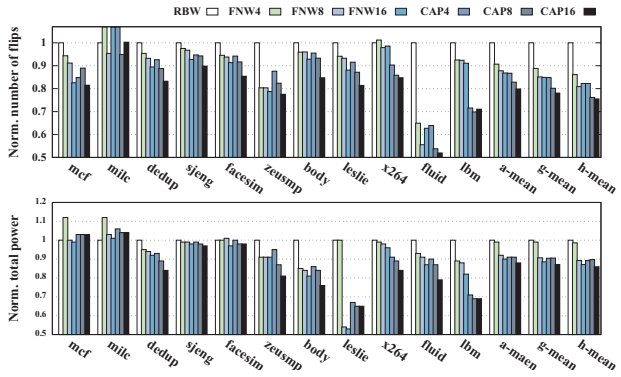

Fig. 4. Normalized bit flips rate and total power consumption of RBW, FNW and Captopril (4, 8 and 16 partitions are considered for FNW and Captopril).

scheme. For example, if storing the even locations in inverted form leads to lower bit flips, a *Minimum Finder* unit selects this scheme for storing the block. Hence, the even locations are inverted and the block is sent to write unit. Since 4 storing scheme are employed, we need 2 bits per block for determining the form of data write. When a **read** is issued, firstly by determining the storing form, we invert the specific locations. Finally, the data block is passed to the requestor.

V. Evaluation

A. Evaluation methodology

We employ gem5 simulator [12] to conduct various experiments. An out-of-order 4-core system with ALPHA ISA is configured in gem5. The detailed area, power and access latency of the PCM memory hierarchy is estimated via NVSim [11]. Three levels of caches and a banked PCM main memory are arranged as the memory system hierarchy. L1 is private cache while L2 and L3 are shared among 4 cores. To enervate the long write latency of PCM memory, a large DRAM cache (16MB) is used. The last level of memory system is a SSD of unlimited size with 25μs response time. The line size of all caches and the main memory is equal to 64B. We select a set of multi-program applications from SPEC-CPU 2006 suite [16] and a set of multi-threaded applications from PARSEC suite [4] for running on gem5. We conduct our simulations for 8 billion instructions with 4 billion instructions for warm-up.

B. Evaluation settings and metrics

We evaluate different forms of *Captopril* in addition to the baseline and a state-of-the-art scheme, FNW [7]. We use *normalized number of bit flips* as a metric for showing bit flips reduction and *total power consumption*. Note that both *Captopril* and *FNW* are based on partitioning. Each partition (word) in our scheme needs 2 bits to indicate the employed rule (00 for RBW, 01 for 2X, 10 for 8X+4 and 11 for LHW). Captopril with LHW-n (CAPn, for short), $n > 32$, imposes more than 25% storage overhead which is not acceptable and CAP1 and CAP2 cannot utilize the benefits of *Captopril*. Hence, we show the normalized number of bit flips and total power consumption results when n= 4, 8 and 16.

C. Evaluation results

Fig.4 shows the normalized number of bit flips and normalized total power consumption of different schemes for different workloads. As can be seen, *Captopril* with n=4, 8 and 16 (CAP4, CAP8, and CAP16) reduces the number of bit flips by 14%, 18% and 21% compared to RBW, respectively. Additionally, Captopril lowers the number of bit flips by 5%, 6% and 9% in comparison to FNW using the same number of partitions. In the best case (CAP16), the number of bit flips is reduced by 21%, on average, while FNW is lowered this metric by 14%. Additionally, this figure shows that increasing the number of partitions is an effective way for reducing bit flips. Unfortunately, we cannot increase number of partitions above 16 due to intolerable storage overhead.

Considering the power overhead imposed by *Captopril* and FNW, Captopril improves over the baseline and FNW by 12% and 4% with its 16-partition configuration. Although in general FNW and *Captopril* do not impose high power consumption overheads to the system, memory intensive applications like `milc` may suffer from this overhead badly.

VI. Conclusion

We introduced *Captopril* based on the observation that positions of frequent bit flips can be predicted using some simple rules. Formulating the locations of these hot bits, we inverted them for write. Full-system simulation results revealed that *Captopril* could reduce the number of bit flips and total power consumption by 21% and 12%, on average, respectively, over the RBW [7] scheme.

References

[1] R. Maddah, et al, *CAFO: Cost aware flip optimization for asymmetric memories*, HPCA, 2015.
[2] M. Jalili, et al, *A Reliable 3D MLC PCM Architecture with Resistance Drift Predictor*, DSN, 2014.
[3] M. Asadinia, et al, *Prolonging Lifetime of PCM-Based Main Memories Through On-Demand Page Pairing*, TODAES, 2015.
[4] C. Bienia, et al, *The PARSEC Benchmark Suite: Characterization and Architectural Implications*, PACT, 2008.
[5] DB. Dgien, et al, *Compression architecture for bit-write reduction in non-volatile memory technologies*, NANOARCH, 2014.
[6] H. Hajimiri, et al, *Content-aware encoding for improving energy efficiency in multi-level cell resistive random access memory*, 2013.
[7] S. Cho, et al, *Flip-N-Write: A Simple Deterministic Technique to Improve PRAM Write Performance, Energy and Endurance*, MICRO, 2009.
[8] J. Chen, et al, *Energy-aware Writes to Non-volatile Main Memory*, SIGOPS Oper. Syst. Rev., 2011.
[9] A. Alsuwaiyan, et al, *MFNW: A Flip-N-Write architecture for multi-level cell non-volatile memories*, NANOARCH, 2015.
[10] PM. Palangappa, et al, *Flip-Mirror-Rotate: An Architecture for Bit-write Reduction and Wear Leveling in Non-volatile Memories*, GLSVLSI, 2015.
[11] X. Dong, et al, *NVSim: A Circuit-Level Performance, Energy, and Area Model for Emerging Nonvolatile Memory*, TCAD, 2011.
[12] N. Binkert, et al, *The Gem5 Simulator*, CAN, 2011.
[13] M. Jalili, et al, *A compression-based morphable PCM architecture for improving resistance drift tolerancee*, ASAP, 2014.
[14] AN. Jacobvitz, et al, *Coset Coding to Extend the Lifetime of Memory*, HPCA, 2013.
[15] XW. Zhang, et al, *WoM-SET: Low power proactive-SET-based PCM write using WoM code*, ISLPED, 2013.
[16] Jl. Henning, et al, *SPEC CPU2006 Benchmark Descriptions*, CAN, 2006.
[17] SM. Seyedzadeh, et al, *PRES: Pseudo-random Encoding Scheme to Increase the Bit Flip Reduction in the Memory*, DAC, 2015.