# Cellular Automata-based Wear Leveling in Resistive Memory

Sutapa Sarkar

*Electronics & Communication Engineering*
*Seacom Engineering College*
*West Bengal, India*
*Email: sutapa321@gmail.com*

***Abstract*—Due to *spatial locality of reference*, repetitive writes cause stress to few contiguous cache memory blocks. If the underlying technology is resistive memory, those blocks suffer with endurance problem and wear out at much faster rate than NAND/NOR Flash memories ($10^5$ to $10^6$ program/erase cycles). In CMPs cache memory, uneven distribution of writes and/or *malicious attacks* cause system failures due to repetitive writes on a particular memory block. To avoid *wear out* of memory cells at premature stage, writes are distributed through *wear leveling* schemes. This work reports an efficient scheme of wear leveling in *resistive memory* using the concept of Cellular Automata ($CA$). Probably this is a unique effort of wear leveling, using Von Neuman's concept of cellular automata, that targets to make uniform writes throughout all memory blocks with *spatial access pattern predictions*. The contiguous *write-stressed* memory blocks are defined as a zone subjected to remapping. Two stage hierarchical Periodic Boundary Cellular Automata ($PBCA$) is used to perform *Density Classification Task (DCT)* to select the cache zone (remapping candidates). Thereafter, identified memory blocks are remapped to new addresses by an *algebraic* technique. This *access aware* write management policy can be worked along with fault tolerant design for implementation of a robust memory subsystem.**

***Keywords-Resistive memory*; *Spatial access characteristics*; *Wear leveling*; *Density classification task*; *Periodic Boundary Cellular Automata*.**

## I. INTRODUCTION

Von Neuman's computing models of pipelining and super-scalar processor [25] [26] become performance limited by reduced throughput. Those processors are also not suitable for having their increased power budget exceeding the standard of Moore's law. Multicore architecture or chip multiprocessors (CMPs) has been considered as the proper replacement of single core architecture by integrating several processors in a single chip. But CMPs requires large *on-chip* cache, which covers an appreciable area of total chip die. Therefore, resistive memory (RRAM/ReRAM) is considered as replacement technology of DRAM/Flash memories for having the advantages of higher package density & scalability with lower energy consumption [1] [2]. But the main disadvantage of ReRAM lies with *write issues* related to endurance and reliabilty problem of resistive memory cells. Due to spatial locality of reference, few blocks are suffering from write stress where most of them are seldom written. In that scenario, *life time* may get hampered at least 20X faster than uniformly distributed writes for worn-out cells [12].

Wear leveling is a scheme that tries to make uniform distribution of writes applied to NAND Flash, as well as NVRAM technologies for the last few decades [11]. It can also be applicable in resistive memory but with little technology specific tunning. Remapping of write request of a memory block to another memory block is the process of achieving wear leveling [6]. *Wear leveling* schemes are categorized at two different levels: 1) cause of multiple writes on the memory blocks having unbalanced distribution of workload of CMPs or for malicious attacks, 2) depending on the technique of look up table based/algebraic method based. Different types of wear leveling schemes are compared in Table I, providing *workload/attack-based* and algebraic/table-based schemes.

*Start gap* scheme redirects write operation from one block to another neighbouring memory block if write count exceeds the threshold limit [3]. Write-stressed data blocks (*remapping candidates*) are found by assessing processor's nonuniform write access pattern to the hybrid memory composed of Static random-access memory (SRAM) & Spin-transfer Torque Magnetic Random-access Memory (STTRAM). Block address remapping is based on an algebraic (randomized) method which inserts a conversion step in between logical address and physical address. The remapping whereabouts of memory blocks are tracked by *Start & Gap* registers. The system lifetime is improved as compared to without *wear leveling* schemes but at the cost of increasing overhead in terms of latency, power and storage. In [4], inter-set and intraset remapping distributes writes between hybrid memory (SRAM & STTRAM). The property of locality of reference of write access is reviewed during selection of remapping candidates. Write counts are checked by cache line & cache set counter. But, the increased number of counters increases hardware overhead, as well as the design complexity.

Typical workload-based schemes are not capable of handling repetitive writes caused by malicious attacks that can also cause system failure within a very short time. *Practical Attack Detector (PAD)* is proposed in [7] to prevent cell failures due to malicious attacks by tracking the write streams within a short time frame. *Rancar* scheme is proposed to handle repeat address attack (RAA) through adaptive remapping in hybrid cache memory [9]. Translation of physical address to intermediate address for intraset/inter-set remapping is performed by swapping set index bits & tag bits. A table based (deterministic method) remapping scheme is proposed in [24] for resistive memory. Algebraic address remapping is proposed in [1] [3] [7] to fit within limited space overhead and to eliminate the requirement of look up table as it's size grows linearly with memory capacity.

Recently, Cellular Automata (CA) has become popular in different applications of cache systems like data migration, protocol processor design, fault tolerance circuit design, etc. [18] [19] [22]. In this paper, a CA-based approach is taken for workload based wear leveling in resistive memory. This research work attempts to increase the system lifetime [3] by enhancing the reliability

TABLE I. COMPARATIVE STUDY BETWEEN DIFFERENT WEAR LEVELING SCHEMES

| Schemes | Memory technology | Granularity | Parameter reviewed | Remapping method | Limitation/Special feature |
|---|---|---|---|---|---|
| *Start-gap* [11] | PCM | Cache line | Spatial write activity | Algebraic method | This scheme is applicable for workload based stresses and malicious attacks. |
| *Rancar* [9] | Hybrid- DRAM & PCM | Cache Set | Repeated set attack | Algebraic method | It affects the spatial locality of reference adversely during remapping. |
| *OWL* [16] | NAND FLASH | Block | Temporal write activity | Table based | The scheme observes flip bits to ensure reduction of overwrites thereby eliminating redundancy of repetitive write operations. |
| *WAPTM* [10] | PCM | Page | Write-activity | Table based | Already used in Google Android 2.3 based on ARM architecture to reduce write activities to page table. |
| *Software based wear-leveling* [15] | Hybrid- PCM+DRAM | Memory address | Write activity | Integer linear programming formulation & polynomial-time algorithm | Hardware requirement is eliminated |
| *PAD* [7] | PCM | Cache line | Malicious attack | Not addressed | It can be effective to other memory technologies apart from PCM to detect malicious attacks by calculating attack density. |
| *Ouroborous* [12] | NVRAM-PCM/FeRAM/STT-MRAM | Local & global region | hybrid - both demand & attack | Hybrid scheme with both table & algebraic method based | It determines access pattern as well as demand prediction for wear leveling. |

of resistive memory cells by redistributing writes. Write-stressed memory blocks as well as less written memory blocks are identified from access pattern.

Write distribution among the memory blocks shows program locality within few contiguous memory blocks termed as zone. The efficacy of any wear leveling scheme depends on the selection of the size of the zone. Here, in this paper, the small-sized source & target remapping zone is identified by parallel operation. Spatial locality of reference is utilized to predict precise and specific write-stressed & less written zone by two stage hierarchical $DCT$ using $PBCA$. An algebraic address translation method is adopted to achieve local/global remapping (address translation of central memory address) of the zone. This simple but cost-effective scheme can be a suitable alternative of workload-based traditional wear leveling schemes [3] irrespective of memory technology.

Section II introduces a relevant theory of CA. CMPs cache architecture is explained with a typical example in Section III. An overview of the design is explained with a block in Section IV. Sections V and VI, describe a CA-based methodology for finding out the remapping candidates. The remapping equation is developed in Section VII. Section VIII deals with analysis of the current work and the paper is concluded with Section IX.

## II. BASICS OF CELLULAR AUTOMATA (CA)

A cellular automaton consists of a number of cells evolving in discrete space and time according to some transition function ($f_i$). It can be viewed as an autonomous Finite State Machine (FSM) which is used to model a variety of physical systems [17] [21]. CA has been accepted as an attractive hardware structure for cache system modelling because of the following characteristics: 1) simple and identical/homogeneous structure, 2) modular, 3) restriction to local interactions among the states of left ($S_{i-1}^t$), centre ($S_i^t$) and right ($S_{i+1}^t$) cells. Each CA stores a discrete variable at time t and it refers to the present state ($S_i^t$) of the cell. In a two-state CA, the values of the states are '0' and '1'. In

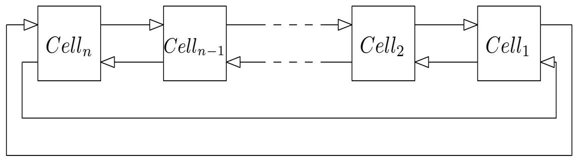

Figure 1. Block diagram of n cell periodic boundary CA.

one-dimensional 3-neighborhood CA, the next state of the $i^{th}$ cell is specified by the next state function ($f_i$) as given by the Eq. 1

$$S_i^{t+1} = f_i(S_{i-1}^t, S_i^t, S_{i+1}^t) \tag{1}$$

CA cell is implemented with D Flip/Flop and a combinational logic circuit realizing $f_i$. Next state of the $i^{th}$ cell is expressed in the form as described in each row of Table II and the decimal equivalent of its NSs is referred as 'Rule' ('232', '184' & '226') as given in Table II. In a 3-neighbourhood CA, there can be a total of $2^8$ (256) rules. The collection of the rules ($R_i$s) applied to each cell of the set form rule vector R = $< R_1, R_2 \ldots R_n >$. For a linear CA, $f_i$ employ only EXOR logic. Nonlinear CA employs logic functions (AND, OR, NOT, etc.). Whenever all the $R_i$s of R are linear/additive, the CA is also referred to as Linear/Additive. A **nonuniform or hybrid CA** uses different rules for individual cells. Uniform CA is a special case of hybrid CA having $R_1 = R_2 = \cdots = R_n$. The state transition diagram shows the sequence of states during its evolution with time called CA behavior, as shown in Figure 8. A state transition diagram may contain single or multiple cycles and on that basis, CA can be categorized as reversible or irreversible CA. If $S_0 = S_n$ and $S_{n+1} = S_1$, the CA is referred to as **periodic boundary** CA, as shown in Figure 1. If $S_0$ = 0 (null) and $S_{n+1}$ = 0, the CA is called **null boundary**. A one-dimensional binary CA is initialized with an IC (Initial configuration) or seed (random in nature) is

TABLE II. TRUTH TABLE

| Present State | 111 | 110 | 101 | 100 | 011 | 010 | 001 | 000 | Rule |
|---|---|---|---|---|---|---|---|---|---|
| RMT | (7) | (6) | (5) | (4) | (3) | (2) | (1) | (0) | |
| Next State | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 232 |
| Next State | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 184 |
| Next State | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 226 |

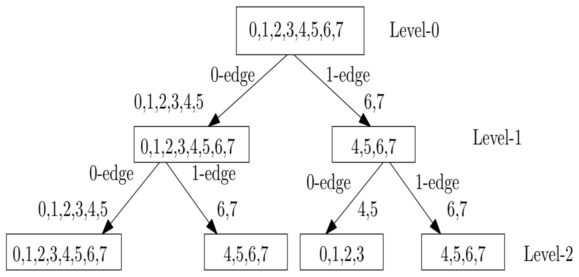

Figure 2. Reachability tree for <232,232,232>.

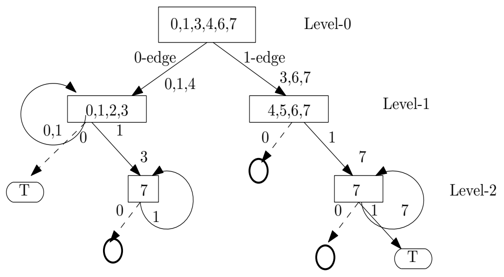

Figure 3. Reachability tree for attractor of hybrid rule <232,184,184,...>.

iterated for a maximum number of steps or until it reaches a fixed point attractor. One-dimensional and two-state $CA$ can classify binary strings according to the densities of (1's or 0's). The ability of a particular elementary CA to solve the density classification task depends on the IC. ICs containing more ones or zeros are closer in hamming distance to one of the solution fixed points or attractors. If IC contains more ones (zeros) than zeros (ones), the CA settles to a fixed point of all ones (zeros). *DCT* may also be performed by realizing two different CA rules hierarchically. First rule is iterated for $n_1$ time steps and second rule is iterated for $n_2$ time steps on the resulting configuration [14]. In solving density classification task using $PBCA$, the following features are required [22]:

$R_1$: All 0s and all 1s two single length attractors

$R_2$: Binary string with more than 50% 1s (0s) fall on all 1s (0s) basin

$R_3$: No other attractors or multilength cycle

The number of reachable states of hybridized rule <232,184,184,184,184,184> is represented by a binary tree called reachability tree, as shown in Figure 3. Root node represents all possible RMTs. 0-edge (1-edge) presents RMTs those have next state value as '0' (1). Reachability tree can also be used to represent cycles/attractors, as shown in Figure 2.

## III. Cache Architecture in Chip Multi-Processors (CMPs)

To avoid bottleneck of off-chip interconnect, CMPs use *on-chip* two/three cache layers (L1 & L2 or L1, L2 & L3) which comprises of private cache (L1) of each core and last level cache (L2/L3) shared among the cores. Resistive memory (ReRAM) is available in variety types as Phase Change memory (PCM), Spin-transfer Torque memory (STTRAM), Magneto Resistive RAM (MRAM), Ferro-electric RAM (FRAM) and Memristors. It stores information in terms of low/high resistance to represent 0/1 logic states. Comparison of different types of memory technology is detailed in Table III. Hybrid cache composed of SRAM and other types of ReRAM are proposed in [1] [4] [5]. Four cores are connected to on-chip hybrid shared Last Level Cache (LLC) as shown in Figure 4. Cores are connected to LLC through bus, switch or a hybrid type of interconnect.

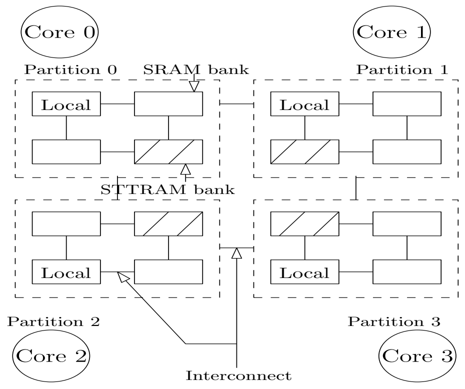

Figure 4. Partitioned hybrid cache architecture with STTRAM & SRAM.

## IV. Block Diagram of Remapping Process

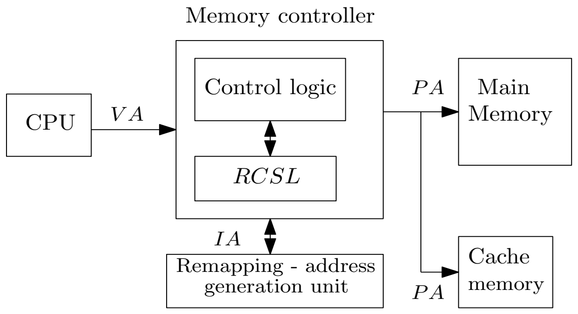

Figure 5. Block Diagram

In Figure 5, block diagram of a memory subsystem with in-built remapping address generation is described briefly. The

TABLE III. COMPARISON BETWEEN DIFFERENT TYPES OF RESISTIVE MEMORY.

| Memory technology | FeRAM | MRAM | STT-RAM | PCM |
|---|---|---|---|---|
| Nonvolatility | Yes | Yes | Yes | Yes |
| Cell size | Large | Large | Small | Small |
| Package Density(ratio) | Low | Low | High | High |
| Read access time(ns) | 20 to 80 | 3 to 20 | 2 to 20 | 20 to 50 |
| Write access time(ns) | 50 | 3 to 20 | 2 to 20 | 20 |
| Write energy consumption | Mid | Mid-High | Low | Low |
| Cell lifetime (in terms of number of writes) | $10^{12}$ | $> 10^{15}$ | $> 10^{16}$ | $10^{12}$ |

shared cache is assumed here with set-associative mapping policy. Cellular Automata based Remapping Candidate Selection Logic (*RCSL*) selects the write-stressed memory blocks with the help of *memory controller* and it is detailed in the next Section. Virtual Address (VA) is generated by the processor at compilation phase and which is translated to Physical Address (PA) by memory controller. But for the case of *write-stressed* memory block, the address is translated to Intermediate Address (IA) by remapping address generation unit. Further, IA is converted to PA for memory operation dictated by memory controller.

## V. REMAPPING CANDIDATE SELECTION LOGIC (RCSL)

In CMPs, multiple processors may access an unpartitioned shared cache memory (L2 or L3) at any instant. Due to *spatial locality of reference*, few contiguous memory blocks are found written for multiple times. Let us assume, M number of processors $P_1$, $P_2$, $P_3$ ... $P_M$ are integrated in a chip with 'N' number of *on-chip* cache lines. An access pattern vector ($AP_v$) is defined to collect write access information of each cacheline.

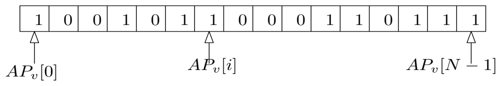

Figure 6. Vector representation of access pattern.

Therefore, length of $AP_v$ is equal to the number (N) of cache lines. If there is a write access for $i^{th}$ block (B[i] say) at any time instant, the value of that vector position $AP_v[i]$='1'or else the value will be '0' as shown in Figure 6. From $AP_v$, the write access of the cache lines are identified. According to write access pattern of memory block $B[i]$, it can be classified into two categories,

1) Write dominated zone and 2) Seldom written zone

Those blocks are seldom or never used can accept a write request redirected from write dominated blocks. When a less written block is found within the memory set itself, *intra-set* memory remapping is done. But when the memory request is transferred to another set, *inter-set* remapping is performed as per Eq. 4 described in Section VIII.

## VI. DESIGN PROCESS

Total memory capacity is divided into groups by taking K number of memory blocks in each. Right/left half of the group (K/2 number of memory blocks) are the remapping zone. Let us take K=6 in this example, as shown in Figure 7. Two stage $DCT$ is performed on $AP_v$ to categorise the blocks. CA is loaded with initial configuration ($IC$ or $seed$) and iterated for appropriate time steps so that it can settle down to an attractor having hamming distance closer to $IC$. The right/left half (three memory blocks) of six contiguous memory blocks are categorized as *write stressed* or *seldom written*.

Each memory block is represented by identical CA cells. 6 cell hybrid $PBCA$ and 3 cell uniform $PBCA$ are applied to perform $DCT$ in two stages. Six-cell hybrid $PBCA$ with uniform rule set < 232 184 184 184 184 184> is iterated for $t_1$ time steps. States are having majority of '0' in their bit pattern, fall in '0' (000000) basin and those having majority '1' fall in basin '63' (111111). Almost uniformly distributed 0's and 1's states fall in alpha ($\alpha$) basin '21' (010101) as given in Table IV. Check bit is the LSB bit of the attractor and saved in a register. For write-stressed block, the checkbit is '1' and zero for less written memory blocks. Therefore, states fall in all ones and alpha basin are both considered as write dominated blocks. Though uniformly distributed 1s and 0s fall on alpha basin violating the requirement $R_3$, it is not inserting any error in the design as the zone may be considered as write stressed.

Let us take an example according to Figure 7. The checkbits of

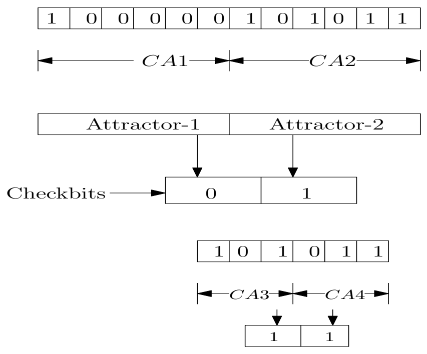

Figure 7. Example-1

CA-1 and CA-2 are found as '0' and '1'. Left half is considered as seldom written zone and right half as write stressed zone. To keep the remapping zone smaller, only three memory blocks is identified from the selected cache zone by making it divide by two. Two groups are iterated to perform $DCT$ again. *3-cell* PBCA

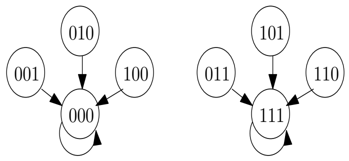

Figure 8. State transition diagram of 3 cell uniform PBCA.

with uniform rule set of $< 232, 232, 232 >$ is iterated for $t_2$ time steps in the second stage. *CA-3* and *CA-4* settle to any of these two attractors '000'/'111' within a single clock cycle as shown in Figure 8. According to the given example, both *CA-3* and *CA-4*

TABLE IV. STATE ANALYSIS REPORT

| | Analysis of basin -'0 | Analysis of basin -'63' | Analysis of basin -'$\alpha$' |
|---|---|---|---|
| States | 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 22, 24, 32, 34, 36, 40, 48 fall in zero basin | 23, 25, 27, 29, 30, 31, 33, 35, 37, 39, 41, 43, 45, 46, 47, 49, 50, 51, 53, 54, 55, 57, 58, 59, 60, 61, 62, 63 | 19, 25, 26, 28, 38, 42, 44, 52, 56 fall in 21 basin |
| Loops | One single length loop 0→0 | One single length loop 63→63 | One multilength loop 21→42→21 |
| Misprediction | 15 (001111) as it falls on '0' basin having greater number of 1s | 33 (100001) as it falls on '63' basin having less number of 1s | No unpredictable states |

settles down to '111' attractor. Checkbit (LSB bit of the attractor) is 1 for LHS as well as RHS. The decision is taken as per decision rule illustrated in the Table V. Therefore, both RHS & LHS are *write dominated* and both are candidates of remapping as per decision rule. LSB bit of alpha basin is '1' and also considered as write-stressed zone. But few states with uniform distribution fall on zero basin. It will contribute a little bit error in the design.

## VII. ADDRESS REMAPPING

Remapping candidates are write-stressed zone (with three contiguous memory blocks) that must be mapped to another less written zone (three contiguous memory blocks). $RCSL$ block identifies the write-stressed zone as well as less written zone through $CA$. Address generation block produces the remapped address by *algebraic* technique in communication with memory controller. *Intra-set* remapping is performed in between blocks of a set and termed as local wear leveling. *Inter-set* remapping or global wear leveling is performed between the sets of a memory.

Set-associative cache memory is studied in the current design where N number of sets with M equal number of blocks to each set is assumed. If G be a non-empty group with a defined operation * in it and H be a subgroup of G. The subset $\{ah : h \in H\}$ is called a left coset of H and denoted by aH. Here aH = Ha (right coset) for all 'a' and so H is a normal subgroup. G & H are represented by ($N_{Totalnumberofmemoryblocks}$, modulo operation) and ($N_{Totalnumberofsets}$, modulo operation). Therefore, the number of coset is given by Eq. 2 [23].

$$[G : H] = \frac{G_n}{H_n} \tag{2}$$

where order or number of elements of G and H are $G_n$ and $H_n$ respectively. The set of distinct cosets or partitions (P) represents the quotient group (Q) and given by the Eq 3.

$$Q = \{\{0 + H\}, \{1 + H\}, \ldots, \{15 + H\}\} \tag{3}$$

Each elements of quotient group (Q) represents the sets (partitions) and the elements belonging to the partitions (sets) are the elements of the sets of *set associative* cache. Therefore, Block address is defined by the Eq. 4.

$$B_i = M * n + j \tag{4}$$

Here, i denotes the block index. Set index (j) and block offset (n) variables are remapping parameters those can be varied from 0 to N-1 and 0 to M-1, respectively, to adjust remapping offsets. To change within set or global wear leveling, change of set index (j) and for local wear leveling, block offset (n) has to be varied in the Eq. 4.

Let us take an example where total number of memory words or length of memory is 256. Number of sets and blocks are taken as 16 (N) & 16 (M). Group of all memory words (G) is $< Z_{256}, + >$ and $< Z_{16}, + >$ represents a subgroup (H). Therefore, the number of cosets is 16 as per Eq. 2. Distinct cosets/partitions are collection of blocks with index numbers are $C_1$ = $<$ 0, 16, 32, 48, 64, ... 240 $>$, $C_2$ = $<$ 1, 17, 33, 49, 65, ... 241 $>$, $\cdots$ $C_{16}$ = $<$ 15, 31, 47, 63 ... 255 $>$. Block of index number 17 can be written as $B_{17}$ = 16*1 + 1. For global/interset mapping, the set index is changed from 1 to 5. So, the new block address will be 21 belonging to set five.

## VIII. ANALYSIS

$SMPCache$ tool is developed to identify memory access traces (opcode read, data read/write) to the data blocks [8] for Symmetric (SMP/DSM) multiprocessors. Table VI illustrates a sample of memory *write access* on the memory blocks from traces of $WAVE$, $NASA7$ and $SWM$ [27]. A typical CMPs architecture with eight cores and three levels of set-associative cache memory are used. To asses the fact of locality of reference, all processor's write access pattern is observed. CEXP, EAR, COMP, MDLJD, HYDRO and others are also considered in Table VII to capture access pattern that is collected from $SMPCache$ simulator. Total memory accesses, number of write accesses and total number of reused memory blocks of all trace files are computed. Observation reveals that distribution of writes are lumped within a very few cache blocks and other blocks are seldom written. So, locality of spatial reference is observed for contiguous memory blocks that can be considered for remapping candidates.

## IX. CONCLUSION AND FUTURE WORK

In this paper, we have proposed wear leveling scheme to reduce write pressure on memory blocks and redistribute write requests within shared LLC of CMPs. Cache lifetime can be improved with local/global wear leveling at the cost of little hardware overhead. But this work addressed the non-uniformity of writes caused by workload distribution of CMPs by considering spatial write access pattern only. Therefore, it is not suitable for those write-stressed memory blocks subjected to malicious attacks. This $CA$ based scheme can be extended for prevention of malicious attacks by capturing temporal as well spatial access patterns as a future work of the current research.

## REFERENCES

[1] I. Lin and J. Chiou," High-Endurance Hybrid Cache Design in CMP Architecture With Cache Partitioning and Access-Aware Policies", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2014.

[2] S. Mittal, Y. Cao and Z. Zhang, "MASTER: A Multicore Cache Energy-Saving Technique Using Dynamic Cache Reconfiguration", *IEEE Transaction on Very Large Scale Integration (VLSI) Systems, 2013.*

TABLE V. DECISION RULE

| Checkbit-0 | Checkbit-1 | Decision | Remarks |
|---|---|---|---|
| 0 | 0 | Neither LHS or RHS is write dominated | Remapping is not required |
| 0 | 1 | RHS is write dominated | 3 cells of RHS are remapping candidates |
| 1 | 0 | LHS is write dominated | 3 cells of LHS are remapping candidates |
| 1 | 1 | Both LHS and RHS are write dominated | Remapping is required for all 6 cells |

TABLE VI. MEMORY BLOCK'S WRITE ACCESS PATTERN

| Block address | Processor | written for single/multiple times |
|---|---|---|
| 3845 | $P_2$, $P_6$ | $P_2$-multiple, $P_6$-single |
| 3844,3843 | $P_2$, $P_5$, $P_6$ | multiple |
| 3596 | $P_0$, $P_4$ | multiple |
| 3141,2045,2037 | $P_1$, $P_3$ | single |
| 2044 | $P_1$, $P_3$, $P_7$ | single |
| 2043 | $P_1$, $P_3$, $P_6$, $P_7$ | $P_1$, $P_7$-multiple & $P_3$, $P_6$-single |
| 2039,2038 | $P_1$, $P_3$, | multiple |
| 2035 | $P_2$, $P_5$ | single |
| 1522 | $P_0$, $P_4$ | single |
| 1521 | $P_0$, $P_2$, $P_4$, $P_5$, $P_6$ | others multiple times except $P_2$ |
| 1520 | $P_0$, $P_4$ | multiple |
| 1519,1518 | $P_0$, $P_4$ | single |
| 1517 | $P_0$, $P_4$ | $P_0$-single, $P_4$-multiple |
| 1516 | $P_0$, $P_4$ | $P_0$, $P_4$-multiple |
| 1515 | $P_0$, $P_4$ | $P_0$-single, $P_4$-multiple |
| 1493, 1416 | $P_0$ | single |

TABLE VII. STATISTICS OF MEMORY BLOCK'S ACCESS

| Trace | Total number of RD/WR Access | Number of Writes | Number of reused blocks |
|---|---|---|---|
| CEXP | 20000 | 262 | 15 |
| WAVE | 3467 | 464 | 27 |
| EAR | 5308 | 262 | 33 |
| COMP | 2524 | 356 | 11 |
| HYDRO | 2127 | 284 | 27 |
| MDLJD | 20000 | 1175 | 31 |
| NASA7 | 1825 | 270 | 21 |
| UCOMP | 2733 | 363 | 11 |

[3] M. K. Qureshi et al. "Enhancing lifetime and security of PCM-based main memory with start-gap wear leveling", "*In MICRO, 2009, pp. 14-23.*

[4] A. Jadidi, M. Arjomand and H. Sarbazi-Azad, "High-endurance and performance-efficient design of hybrid cache architectures through adaptive line replacement", *IEEE, 2011.*

[5] J. Li, C. Xue and Y. Xu, "STT-RAM based Energy-Efficiency Hybrid Cache for CMPs", *IEEE/IFIP 19th International Conference on VLSI and System-on-Chip, 2011.*

[6] L. P. Chang, "On efficient wear leveling for large-scale flash-memory storage systems", *in Proceedings of the 2007 ACM symposium on Applied computing. ACM, 2007, pp. 11261130.*

[7] M. K. Qureshi, A. Seznec and L. A. Lastras, "Practical and Secure PCM Systems by Online Detection of Malicious Write Streams", *17th International Symposium on High Performance Computer Architecture, 2011.*

[8] M. A. Vega Rodr guez, J. M. S Prez, R. Mart n de laae Monta a, and F. A. Zarallo Gallardo, "Simulation of Cache n Memory Systems on Symmetric Multiprocessors with Educational Purposes.", *International Congress in Quality and in Technical Education Innovation,volume 3, pages 4759, 2000.*

[9] G. Wu, H. Zhang, Y. Don, and J. Hu, "CAR: Securing PCM Main Memory System with Cache Address Remapping", *18th International Conference on Parallel and Distributed Systems, IEEE, 2012.*

[10] T. Wang, D. Liu, Z. Shao and C. Yang, "Write-activity-aware page table management for PCM-based embedded systems", *17th Asia and South pacific conference, IEEE, 2012.*

[11] M. K. Qureshi, M. Franchescini, V. Srinivasan, L. Lastras, B. Abali and J. Karidis, "Enhancing Lifetime and Security of PCM-Based Main Memory with Start-Gap Wear Leveling", *in Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture ACM.*

[12] Q. Liu and P. Varman, "Ouroboros Wear-Leveling: A Two-Level Hierarchical Wear-Leveling Model for NVRAM", *IEEE*, 2015.

[13] C. Stone and L. Bull, "Solving the Density Classification Task Using Cellular Automaton 184 with Memory", *2009.*

[14] H. Fuks, ”Solution of the density classification problem with two cellular automata rules”, *Phys. Rev. E 55, R2081(R) Published 1 March 1997.*

[15] J. Hu, M. Xie, C. Pan, C. J. Xue, Q. Zhuge and E. H. Sha, ”Low Overhead Software Wear Leveling for Hybrid PCM + DRAM Main Memory on Embedded Systems”, *IEEE Transactions on Very Large Scale Integration (VLSI), 2015.*

[16] C. Wang and W. Wong, ”Observational Wear Leveling: An Efficient Algorithm for Flash Memory Management”, *2012.*

[17] C. Lee, ”Synthesis of a Cellular Universal Machine using 29 state Model of von Neumann ”, *Automata Theory Notes, The University of Michigan Engineering Summer Conferences*, 1964.

[18] M. Dalui and B. K. Sikdar, ” A cellular automata based self-correcting protocol processor for scalable cmps”, *Microelectronics Journal 62 (2017) 108119.*

[19] S. Sarkar, M. Saha and B. K. Sikdar, ”Multi-bit fault tolerant design for resistive memories through dynamic partitioning”, *EWDTS, IEEE Computer Society, Novisad, Serbia, 2017, pp. 16.*

[20] J. Thatcher, ”Universality in Von Neumann Cellular Automata.”, *In Tech Report 03105-30-T,ORA, University of Michigan.*, 1964.

[21] J. V. Neumann., ”The theory of self-reproducing Automata.”, *In Tech Report 03105-30-T,ORA, University of Michigan.*, 1964.

[22] B. Das, M. Dalui, S. Kamilya, S. Das and B. K. Sikdar, ”Synthesis of Periodic Boundary CA for Efficient Data Migration in Chip-Multiprocessors”, *IEEE, 2013.*

[23] J. Nicholoson, ”The Development and Understanding of the Concept of Quotient Group”, *HISTORIA MATHEMATICA(1993), 68-88*

[24] W. Wen, Y. Zhang and J. Yang, ”Wear Leveling for Crossbar Resistive Memory”, *In Proceedings of the 55th Annual Design Automation Conference DAC ’18, ACM.*

[25] K. Asanovic, J. Beck, B. Irissou, B.Kingsbury and J. Wawrzynek, ”A Single-Chip Vector Microprocessor with Reconfigurable Pipelines”, *In the Proceedings of the 22nd European Solid-State Circuits Conference, Sept. 1996.*

[26] N. N. Sirhan and S. I. Serhan, ”Multi-core Processors: Concepts And Implementations”, *International Journal of Computer Science & Information Technology (IJCSIT), Vol 10, No 1, February 2018.*

[27] M. A. Vega-Rodrguez, J. M. Snchez-Prez, R. M. de la Montaa and F. A. Zarallo-Gallardo, ”Simulation of Cache Memory Systems on Symmetric Multiprocessors with Educational Purposes”, *Proc. of the I International Congress in Quality and in Technical Education Innovation, vol. III, pp. 47-59.*