Periodic Boundary Cellular Automata Based Wear Leveling for Resistive Memory

Sutapa Sarkar, Manisha Ghosh, Biplab Kumar Sikdar, Mousumi Saha

***Abstract*—Locality of reference is preferable every time while writing to cache blocks. The write process may repeat only on a few adjacent cache blocks. As a result, it creates a stress on those blocks. If resistive memory is chosen to be the fundamental technology for the write purpose, the write sensitivity on those blocks increases more. It causes a loss to the durability of the memory. It can be damaged earlier in comparison to NAND/NOR Flash memories (10^5 to 10^6 program/erase cycles). This non-uniformity in writes as well as malicious attacks in CMPs cache can cause sudden breakdown of those systems. The wear out of memory blocks at their primary stages can be avoided by wear leveling through distribution of writes to different blocks.**

This work represents an efficient scheme of wear leveling applied specifically for resistive memories. The major part of this work is developed around one dimensional two state CA. It has the aim to achieve uniform writes throughout all memory blocks with spatial access pattern predictions. The adjacent write-overloaded blocks are considered as an area subjected to remapping. The periodic boundary cellular automata (PBCA) employed for the scheme performs density classification task (*DCT*) to choose the remapping zone of memory. Remapping is executed through changing the current address location of the identified memory blocks to the new location. Further, the proposed memory write management policy can be implemented together with the fault tolerant memory architecture to develop a memory subsystem with more durability and robustness.

***Index Terms*—Spatial access characteristics; Wear leveling; Density classification task; Periodic Boundary Cellular Automata.**

I. Introduction

THE performance of computing models of pipelining and superscalar processor [1] [2] is limiting due to reduction in throughput. Those processors require an increasing power budget which exceeds the standard of Moore's law. So, these are losing their suitability now a days. Chip multiprocessors (CMPs) are on the other hand the combination of several single processors fabricated in a single chip. Single core architecture is replaced by multicore processor (CMPs) [3] [4] [5]. But *on-chip* cache size has to be large in CMPs and it acquires a significant area of total chip floor. Therefore resistive memory (RRAM/ReRAM) is chosen in place of DRAM/Flash memories to take the advantages of higher packaging density as well as increasing scalability with lower energy consumption [6] [7]. However, the major disadvantage faced with *write issues* in ReRAM is that it is related to durability and reliability of the memory cells. Locality of reference generates write stress on some memory blocks which are rewritten a large number of times. It causes a severe reduction in *life time* of such memory and that is at least 20X faster than that of uniformly written but worn-out memory cells [8].

Since the last few decades the wear leveling scheme has been used to bring uniformity in the writes to all the blocks of a memory. It has been found to be applied in NAND Flash, as well as NVRAM technologies [9] [11] [33]. Resistive memory writing policy can be improved with this scheme but it requires some technology based modifications. Wear leveling can be defined by the process of shifting a write request from a memory block to another memory block [10]. *Wear leveling* schemes are subdivided into two different parts : 1) cause of multiple writes on the memory blocks having unbalanced distribution of workload of chip multiprocessors (CMPs) or for malicious attacks, 2) depending on the techniques of remapping like a look up table based/algebraic based method. Some categories of wear leveling schemes are compared in Table I, providing *workload/attack-based* and algebraic/table-based schemes.

Typical workload-based schemes are incapable of dealing with repetitive writes due to malicious attacks. Even system failure can occur within a small time span. A scheme called *Practical Attack Detector (PAD)* is proposed in [12] which can stop the cell failure due to malicious attacks by keeping a track on the write streams within a short time frame. The *Rancar* scheme deals with repeat address attack (RAA) through adaptive remapping in hybrid cache memory [13]. It translates the physical address to intermediate address for intraset/inter-set remapping by interchanging set index bits and tag bits. A table based (deterministic method) remapping scheme reported in [14] for resistive memory. Algebraic address remapping, defined in [6] [12] [15] can be adjusted in limited space overhead and to remove the look up table requirements because of the growing size of Look up table shows linearity with memory capacity.

Recently, cellular automata (CA) is being applied to some applications in cache system design such as detection of fault, data migration, protocol processor, fault tolerant and self-corrected memory etc. [16] [17] [18] [19] [20]. In this paper, a *CA* based approach is applied for workload based wear leveling in resistive memory. This research work targets to increase the system lifetime [15] by enhancing the durability of resistive memory cells by uniformly distributing writes in all the blocks. Access pattern identifies the write-stressed memory blocks as well as the less written memory blocks.

Write distribution among the memory blocks shows pro-

Manuscript received May 05, 2019; revised June 11, 2019.

Sutapa Sarkar is with the department of Electronics and Communication Engineering, Seacom Engineering college, Kolkata, India, sutapa321@gmail.com.

Manisha Ghosh is with the department of Electronics and Communication Engineering, NSHM Knowledge Campus, Durgapur, India, manishaghosh.uit@gmail.com.

Biplab K. Sikdar is with the department of Computer science and Technology, Indian Institute of Engineering Science and Technology, Shibpur, India, biplab@cs.becs.ac.in.

Mousumi Saha is with the department of Computer Science and Engineering, National Institute of Technology, Durgapur, India, msaha.nitd@gmail.com.

TABLE I: COMPARATIVE STUDY ON DIFFERENT WEAR LEVELING SCHEMES

Schemes	Memory technology	Granularity level	Parameter reviewed	Remapping technique	Special feature
Start-gap [9]	PCM	Cache line	Repetitive write activity due to spatial locality of reference	Address space randomization using Invertible Binary matrix and Fiestel network based algorithm	Start gap is basically intended for uneven workload distribution but Region based start gap is also effective for malicious attacks.
Car [13]	Hybrid- DRAM & PCM	Cache Set	Repeated set attack	Randomized cache address remapping method by swapping set and tag index bits	It adversely affects the spatial locality of reference for ordinary programs which are not subjected under attacks.
OWL [28]	NAND FLASH	Block	Temporal write activity	Look up table based method	The scheme observes flip bits to ensure reduction of overwrites thereby eliminating redundancy of repetitive write operations.
XWL [14]	Crossbar ReRAM	Block	Effective write activity for different data pattern and row addresses.	Look up table based method	The scheme is based on lifetime of a ReRAM cell as a function of IR voltage drop and large sneaky currents.
WAPTM [24]	PCM	Page	Write-activity	Look up table based method	Already implemented for Google Android 2.3 based on ARM architecture for reduction of write activity into page table.
Software based wear-leveling [27]	Hybrid-PCM+DRAM	Memory address	Write activity	Integer linear programming formulation & polynomial-time algorithm	Requirement of hardware is eliminated
PAD [12]	PCM	Cache line	Malicious attack	Not addressed	Adaptive wear leveling scheme is proposed in addition to prevention of malicious attack by calculating attack density
Ouroborous [8]	NVRAM-PCM/FeRAM /STT-MRAM	Local & global region	Spatial write pattern is observed for access pattern prediction & malicious attack	Hybrid scheme with both table & algebraic based method.	It determines access pattern as well as demand prediction for local and global wear leveling.

gram locality within few adjacent memory blocks mentioned as zone. The appropriateness of any wear leveling scheme lies in the selection of the size of the zone. Here, in this paper, the small-sized source as well as target remapping zone is identified by parallel operation. Spatial locality of reference is applied to predict precise and specific write-stressed and less written zone by two stage hierarchical design using density classification task (DCT) that employs periodic boundary cellular automata ($PBCA$). An algebraic address translation procedure is used to obtain local/global remapping (address translation of central memory address) of the zone. This can be considered as a simple but cost-effective scheme. This scheme can be implemented as an alternative method of workload-based traditional wear leveling schemes [15] irrespective of memory technology.

Section II gives basic concepts of *CA*. Motivation of this research work is introduced in Section III. Section IV describes the design framework of wear leveling in resistive memory. In Section V, *PBCA* based design methodology to find out the blocks to be remapped is explained. The address remapping technique is established in Section VI. Section VII evaluates the performance of the proposed scheme and ultimately the conclusion and future scope is stated in Section VIII.

II. Basics of Cellular Automata

Cellular Automata (CA) can be appropriately used to model physical systems [29] [31] as of cache memory of chip multiprocessors (CMPs). *CA* has the following flavour for modelling and designing of any physical or biological system: i) homogeneous cellular structure, ii) scalability, iii) modularity, iv) local interaction among cells provides global condition of CA, iv) interaction may be confined in between the left neighbouring cell, cell itself and right neighbouring cell and v) parallel processing (parallelism).

The basic architecture of CA is analogous to autonomous Finite State Machine (FSM). So, the cells of a CA are allowed to evolve in discrete space and time. The next state computation of cells is occurred in accordance with applied transition function (f_i) - that is based on rule or logic. A CA cell always stores a value (or state) at discrete time instant 't'. The state of the i^{th} cell (S_i^t) at time t, is referred as the current or present state (PS). Two-state CA can keep only binary values (or states) '0' and '1'. The transition function f_i of a rule in 3-neighborhood, two-state and one-dimensional CA can be given by Equation 1. The next states (NSs) of the i^{th} cell, it's left and right neighbours are specified by S_i^{t+1}, S_{i-1}^{t+1} and S_{i+1}^{t+1} respectively.

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

The cells of CA can be constructed by D flip-flop (Delay flip flop). A digital circuit of combinational logic can be used to realize the logic function f_i in accordance with the applied design rule [30]. Implementation of Rule 232 and 226 in CA is given in Figure 2. The combinational logic functions and D flip-flops are used to design the hardware of hybrid PBCA as shown in Figure 1.

The possible combinations of next states (NSs) of the i^{th} cell of a CA are exemplified in the rows of Table II which cumulatively represent a rule. The rule is expressed in the

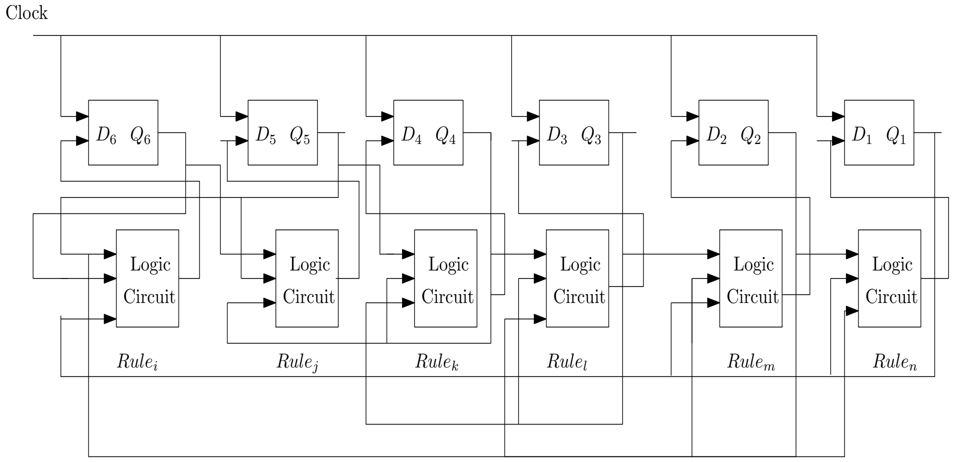

Fig. 1: Block diagram of n-cell periodic boundary CA

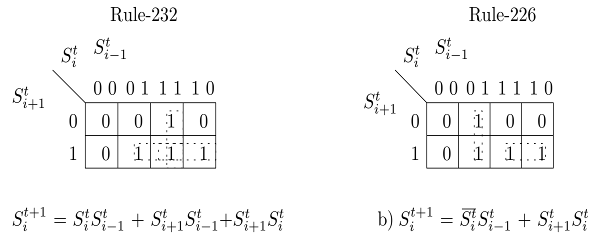

Fig. 2: Next state functions of rules 232 and 226

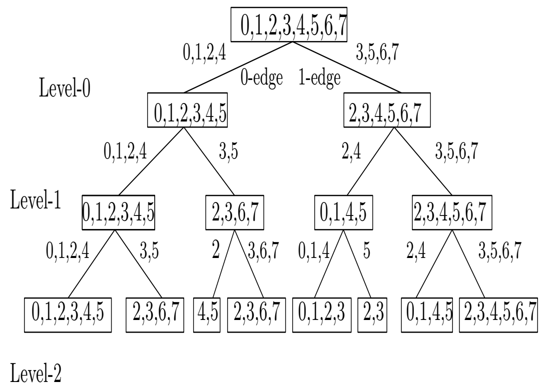

Fig. 3. Reachability tree for <232 232 232>

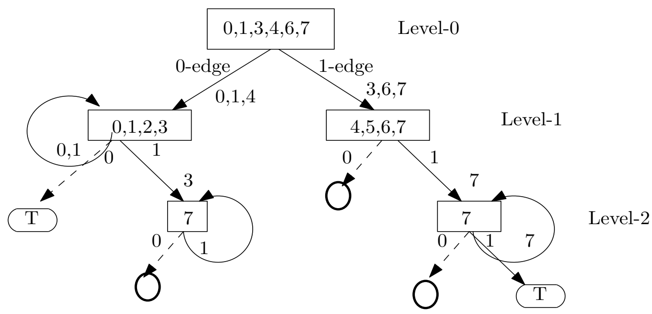

Fig. 4. Reachability tree for attractor of hybrid CA $<$ 232 184 184 ...$>$

format of it's decimal equivalent. For a 3-neighborhood CA, total number of possibility of CA rules are 2^{2^3} (256). Table II shows few of those rules like '232', '226', '192' and '184' which are employed in the current design.

The collective combination of the applied rules (R_is) of a CA represents the rule vector of that CA. A typical example of a rule vector is given in Equation 2.

$$R = < R_1, R_2, \ldots, R_n > = < 232, 184, 184, 184, 184, 184 > \quad (2)$$

In null boundary CA, the status of the left to leftmost cell S_0 is considered as logic value '0' (null) and it is also the case of right of a right most cell S_{n+1} that is '0' (null). For periodic boundary CA, $S_0 = S_n$ and $S_{n+1} = S_1$, as given in Figure 1. A nonuniform or hybrid CA is configured with different rules for it's cells. But, uniform CA which is a special type of nonuniform CA, have $R_1 = R_2 = \cdots = R_n$. CA behaviour can be observed by state transition diagram (STD) which shows the sequence of states for evolution with time as shown in Figure 15. Single or multiple cycles are observed in STD. The CA can be categorized as reversible CA or irreversible CA according to the presence of those cycles.

Reachability tree is a form of binary tree mainly used to represent the reachable states of a CA in different levels of operation. Root node carries all possible RMTs (rule mean terms) of a rule. Those RMTs having '0' as their next states are kept under left edge (0-edge) of the tree. Those RMTs with their next state value as '1' are kept under right (1-edge) of the tree. The number of levels of the tree is determined by the number of cells of a CA.

Rechability tree can also be constructed to detect the possible cycles or attractors formed in the state transition diagram for a rule vector of a CA. Figure 3 shows the reachabilty tree for the uniform CA<232 232 232>. On the other hand, the reachability tree for attractor of hybrid CA$<$ 232 184 184 $\ldots >$ is shown in Figure 4. Here periodic boundary condition is considered for both the cases. Rule 232 and rule 184 are used to realize our wear leveling method for resistive memory.

III. Motivation

With increasing number of cores in CMPs, off-chip memory puts limitation in terms of bandwidth, latency and speed of operation.

To get rid of this bottleneck, on-chip memory is given

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	1	1	0	0	0	1	0	226
Next State	1	1	0	0	0	0	0	0	192
Next State	1	0	1	1	1	0	0	0	184

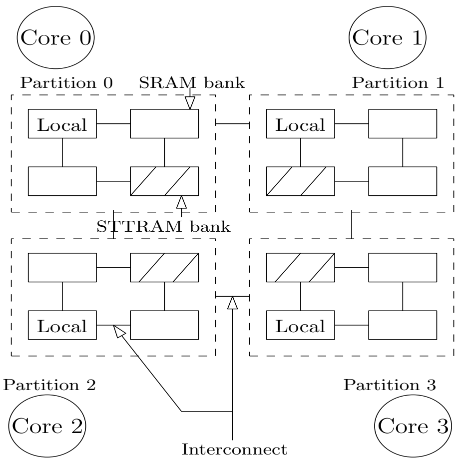

Fig. 5. Partitioned hybrid cache architecture with STTRAM & SRAM

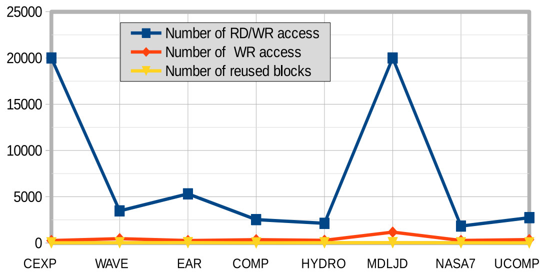

Fig. 6. Statistics of memory block's access in trace files collected from SMPcache

priority over off-chip memory. CMPs generally have *on-chip* two or three cache layers (L1 and L2 or L1, L2 and L3), from which L1 is private cache of each core and L2 or L3 is last level cache shared among the cores. But a large-sized on-chip cache is required to mitigate the demand of the growing number of cores. The cache can be accommodated within the chip area if it can have higher package density and scalability like resistive memory. ReRAM is available in varieties namely Phase Change memory (PCM), Spin-transfer Torque memory (STTRAM), Magneto Resistive RAM (MRAM), Ferro-electric RAM (FeRAM), Memristors etc. Different types of resistive memory technology is compared in Table III. Hybrid *on-chip* cache has also been proposed in [6] [21] [22] where resistive memory and SRAM/DRAM are used together to exploit the advantages of both. A typical CMPs with four cores and shared Last Level Cache (LLC) is shown in Figure 5. Cores may be interconnected to hybrid LLC through bus, switch or a hybrid type of connectivity.

SMPCache simulator is developed to consider uniprocessor or multiprocessor traces which represent the memory access in terms of opcode read, data read and data write used for SMP/DSM multiprocessors. Single processor traces are considered for the current work that are taken from SPEC92 benchmarks (Hydro, Nasa7, Cexp, Mdljd, Ear, Comp, Wave, Swm and UComp). These are the few example traces collected from some real tests carried out on a MIPS R2000 system. All these represent a wide variety of "real" application programs which are come from the Parallel Architecture Research Laboratory, New Mexico State University.

A typical eight-core CMPs with two levels of set-associative cache architecture are configured for simulation to identify memory accesses (opcode read, data read and write) to the memory blocks. [23]. Figure 6 shows the statistics of total number of accesses, write accesses and number of repetitively written memory blocks for $WAVE$, $NASA7$, SWM etc [23]. To assess the fact of spatial locality of reference, all processor's (P_0-P_7) write access patterns are observed and detailed in Table IV by complete system simulation of *SMPCache*. Observation reveals the fact that the write-distribution is concentrated within a very few number of cache blocks and other blocks are seldom written. Further, spatial locality of reference is noticed for contiguous memory blocks (later termed as cache zone) that can be considered for remapping candidates. Therefore, the current work is motivated to investigate a solution for resistive memory that ensures reliable write operation for a robust memory system through wear leveling. Algebraic remapping technique is considered to increase the cell lifetime that makes a high endurance memory system.

IV. Design Framework

To adopt wear leveling in ReRAM/RRAM, two distinct types of operations need to be performed: i) Selection of source (write stressed) and target (seldom written) memory blocks to be remapped and ii) Determination of the target address from the source address. In this design, CMPs are assumed to have two layers of *on-chip* cache (L1 and L2), where L1 is private to each core and shared L2 is considered with set-associative mapping policy. Logical, contiguous and fixed-size portion of memory are considered as remapping zone. Remapping zone is found from write access pattern of memory block by employing *DCT*.

The Last level cache (LLC) blocks are considered into groups with M number of memory blocks per group. In pre-processing stage, these groups are categorized into two types: i) seldom or never used and ii) write dominated. Further, at decision stage, right and/or left half of the group (M/2 blocks) are selected to get the exact remapping candidates. Hierarchical with two stage *CA* based density classification task (*DCT*) is performed on the access pattern of the memory blocks to identify the category of memory blocks. Global/local remapping is performed by redirecting the write request. When less written zone is found within the

TABLE III. COMPARISON ON DIFFERENT RESISTIVE MEMORY TYPES

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}

TABLE IV. WRITE ACCESS PATTERN

Processor	Block address	Number of times written
P_0	1416, 1493	single
	1515-1517, 1518-1520, 1521-1522	1516, 1520-1521 multiple
	3596	multiple
P_1	2037-2039	2037-single
	2043-2045	2043-multiple
	3141	single
P_2	1521	single
	3843-3845	3845-single
P_3	2037-2039	2037-single
	2043-2045	single
	3141	single
P_4	1515-1517	multiple
	1518-1520	1520-multiple
	3141	single
P_5	1521	multiple
	2035	single
	3843-3844	multiple
P_6	1521	multiple
	2043	multiple
	3843-3845	3845-single
P_7	2043-2044	2043-multiple
	3843-3845	3845-single

same memory set, local or *intra-set* remapping is performed. But when it is found in different set, global or *inter-set* remapping is performed.

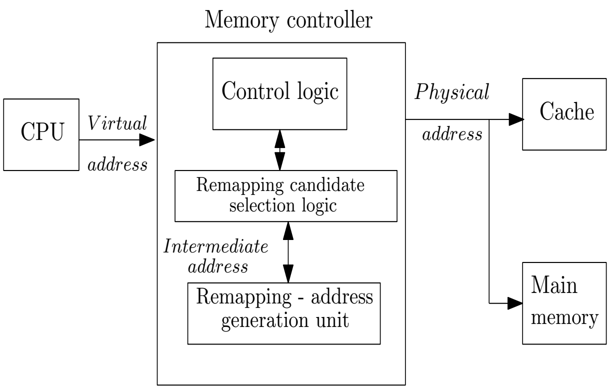

Fig. 7. Remapping in memory sub-system

Block diagram of a tentative design frame work with memory subsystem is described in Figure. 7. Processing unit generates virtual address which is further translated into physical address for memory access. But remapping technique creates a translation level in-between virtual address and physical address. Therefore, memory controller is equipped with sub-block control logic, remapping candidate selection logic and remapping address generation unit. Remapping Candidate Selection Logic (*RCSL*) is a *CA* based unit used to detect the write-stressed memory blocks (source zone) as well as less written memory blocks (target zone). Those memory blocks which do not require any address translation can directly be applied to physical memory. But when remapping is required, memory controller sends the control to address generation block where the central address of source zone is translated to target address (central address of target zone). So, the virtual address is translated to physical address through intermediate address for accessing the physical memory as shown in Figure 7.

A. Density classification task

The CA can be initialized with a binary pattern called Initial Configuration (IC) or seed. The seed is random in nature [25]. After that, CA is allowed to iterate up to a permissible number of steps. It may reach an attractor which does not allow further changes of states anymore. One-dimensional and two-state *CA* can be used to discriminate binary strings according to the densities of (1s or 0s). This operation is called density classification task (DCT).

The CA has the capacity to classify the initial configurations (ICs) having more ones or zeros in their bit pattern. They use to settle to an attractor having minimum hamming distance from the attractors. If seed contains more 0s (1s) than 1s (0s), the CA settles to a fixed point of all 1s (0s). *DCT* may be performed by realizing one or two stages. In two-stage *DCT*, two different CA rules are applied one after another, hierarchically. First stage with a given rule is iterated for t_1 time steps. The resulting configuration is iterated for t_2 time steps with another rule in the next stage [26].

B. Requirements of PBCA rules for DCT

For PBCA based solution of the DCT applied for the current design, CA has the following required features:

- $Req1$: Two attractors with single length cycle
- $Req2$: Formation of attractors having all 0s and all 1s
- $Req3$: Binary string (seed) having greater than 50% 1s (0s) should fall on all 1s (0s) attractors
- $Req4$: CA does not contain any other attractors (single length or multilength cycle)

C. Selection of design rules

For two-stage operation, two different sets of rules are selected. Six cell hybrid *PBCA* with rule vector $<$ 232 184 184 184 184 184 $>$ or $<$ 232 226 226 226 226 226 $>$ can be selected for the first stage. Three cell uniform rule vector is found appropriate for decision stage Figure 7.

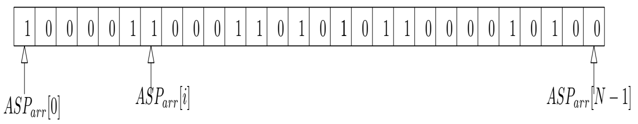

Fig. 8. Access pattern array

As per the given STD of 6-cell hybrid $<$ 232 226 226 226 226 226 $>$ or $<$ 232 184 184 184 184 184 $>$, $Req1$ and $Req2$ are satisfied for the following design. As, there are two attractors (all 0s and all 1s) in PBCA rule employed in the current design. Further, all 0s and 1s attractors are single length cycle attractors those present in STD as shown in Figure 9 or Figure 14. But it has another attractor defined as α-basin which is a multi-length cycle attractor. Though it does not contribute any error as it is interpreted in another way. Decision stage is based on rule $<$ 232 232 232 $>$ for the presence of all 0s and all 1s single length cycle attractors.

V. Design methodology

CMPs have many processors integrated in a single chip die. Those processors may access a contiguous shared cache memory (L2) at any instant. Due to uneven distribution of work-load, few memory blocks are written repetitively. These blocks are contiguous in nature as *spatial locality of reference* is observed in their access pattern. Let us assume, k number of processors (cores) Q_1, Q_2, Q_3, ... , Q_k are integrated within a chip. An access pattern array (of length N) keeps the write access information of each and every last level cacheline whose length is also 'N'. Spatial access pattern register (ASP_{arr}) keeps the information of write access in terms of binary nonzero value. At any time instant, if the i^{th} block $MB[i]$ is written, the position of ASP_{arr}[i] is set as shown in Figure 8.

Each memory block can be represented by identical CA as it stores binary values, those are updated at discrete time instants. CA is iterated upto an attractor state or stable state. Here, CA settles down to an attractor basin having hamming distance closer to initial configuration (seed). Six cell hybrid CA with rule vector $<$ 232 184 184 184 184 184 $>$ is iterated upto the depth of CA or until it reaches an attractor of 0-basin or 1-basin. LSB of the attractor are considered likewise:

- Case-1: Lsb of CA is 0 for seldom written zone
- Case-2: Lsb of CA is 1 for write dominated

State transition diagram of 6-cell hybrid PBCA with rule vector $<$ 232 184 184 184 184 184 $>$ is shown in Figure 14. 3-cell uniform PBCA $<$ 232 232 232 $>$ is employed in decision stage which is carried out for single time step. Check bits (LSB bits of CAs) are collected and saved to infer results.

To keep the remapping zone small, M=6 is assumed in this example as shown in Figure 10. Therefore, number of groups are formed having six cells in each. The first stage and final stage are operated hierarchically to perform *DCT* for categorization of memory blocks. CA is loaded with $seed$ taken from ASP_{arr} and iterated. States with more than 50% of '0' ('1') in their bit pattern, fall in 0-basin (1-basin).

Space time evolution of a typical example is shown in Figure 11 (Figure 12). The *CA* falls on all 1s basin in first stage after four (five) clock states. In final stage, it is concluded that left half is *seldom written* zone and right half is *write stressed* zone as per Figure 13.

The left and right half (three memory blocks) are splitted out of six contiguous memory blocks and are processed in final stage (decision phase). 3-cell uniform PBCA with rule $<$ 232 232 232 $>$ is applied in each group. Space time evolution shows every iteration upto the settling points or attractors ('0' and '1') as shown in Figure 13.

Like first stage here also, the check bit is the least significant bit (LSB) bit of the attractor and saved in a register. For write-stressed block, the checkbit is '1' and '0' for less written memory blocks. Here in this example, as shown in Figure 13, left half has fallen in 0- basin and right half has fallen in 1-basin. They are considered as less written or write dominated blocks accordingly. In Figure.10, the generation of checkbits are illustrated. Both the CAs are initialized with the values of access pattern array to find the appropriate remapping zone.

In the first stage (pre-processing stage), alternate rule 226 can be used instead of rule 184. Therefore, 6-cell hybrid CA $<$232 226 226 226 226 226 $>$ can work appropriately as an alternative to $<$232 184 184 184 184 184 $>$. Comparative state analysis report is furnished in Table VIII which shows some minor differences in performance as follows:

i) Depth of CA is different.
ii) Same state may not settle to same attractor.
iii) State transition diagram shows different iteration time for same state transition.

In Figure 10, two groups are considered as the number of memory blocks are 12. The groups correspond to CA-1 and CA-2. The attractor-1 and attractor-2 produce checkbits, which are found as '1' and '0'. Therefore, group-2 is considered as seldom written zone and group-1 is write stressed zone. To keep the remapping zone smaller, only three memory blocks are identified from the selected cache zone by making it divide by two.

These two groups (group-2L and group-2R) are iterated again to perform DCT with the *3-cell* PBCA with uniform

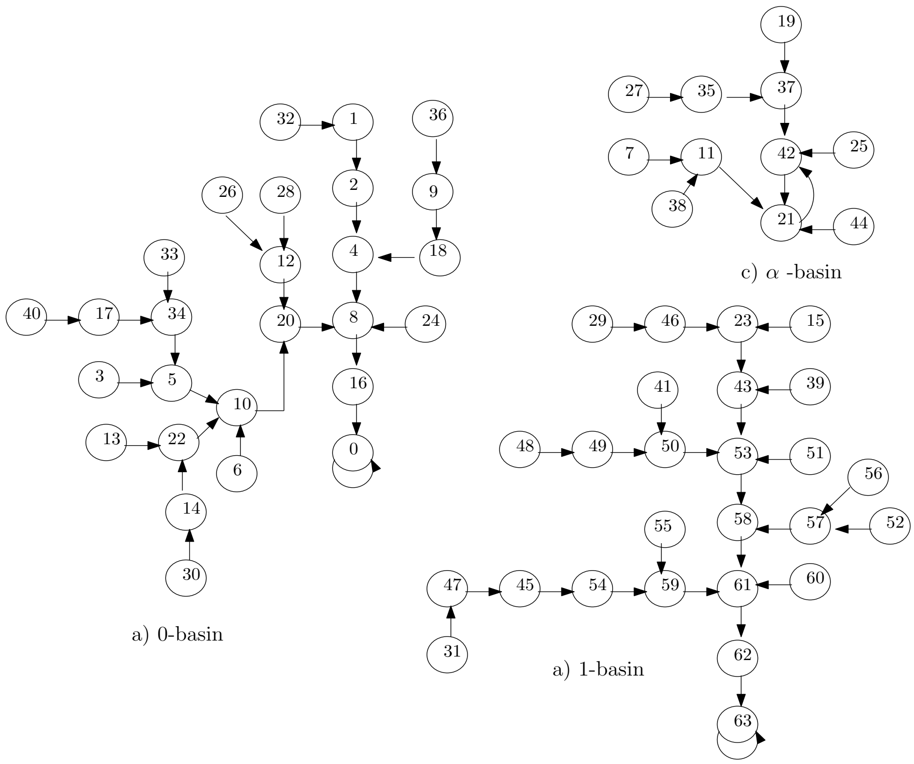

Fig. 9. STD for 6-cell $< 232\ 226\ 226\ 226\ 226\ 226 >$ hybrid PBCA

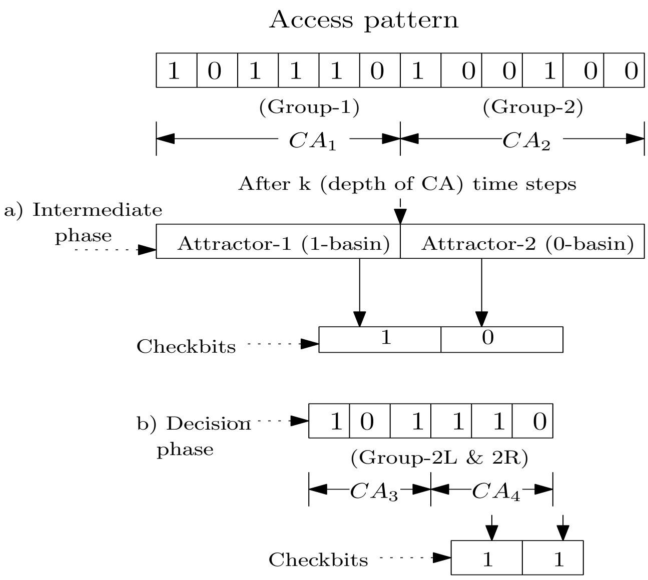

Fig. 10. Generation of checkbits

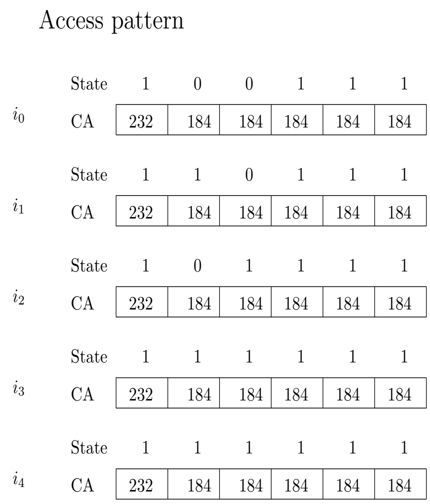

Fig. 11. Space time evolution of pre-processing stage with hybrid rule $< 232\ 184\ 184\ 184\ 184\ 184 >$

rule set of $< 232\ 232\ 232 >$. *CA-3* and *CA-4* settle to any of the two attractors '000' or '111' within a single clock cycle as shown in Figure 15. According to the given example, both *CA-3* and *CA-4* settle down to '111' attractor. The checkbit (LSB of the attractor) is '1' for left hand side as well as the right hand side. The decision is taken as per decision rule as illustrated in Table V. Therefore, both RHS and LHS are *write dominated* and the candidates of remapping as per decision rule.

VI. Address remapping technique

Remapping candidates are write-stressed zone (three contiguous memory blocks) that must be mapped to another less written zone (three contiguous memory blocks). Remapping Candidate Selection Logic ($RCSL$) block identifies the write-stressed zone as well as less written zone through CA. *Intraset* remapping is performed in between blocks of a set (local wear leveling) and *Interset* remapping is performed among different sets (global wear leveling) [8]. The source zone (write-stressed zone) of remapping are found alongwith the target zone (less written zone) in parallel operation. The conversion of block address from write-stressed zone to seldom written zone is performed in *remapping-address generation unit*. New block address generation (intermediate address) from older ones is achieved through *algebraic* technique for local (*Intraset*) as well as global *Interset* zone remapping. Address generation block produces the remapped

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

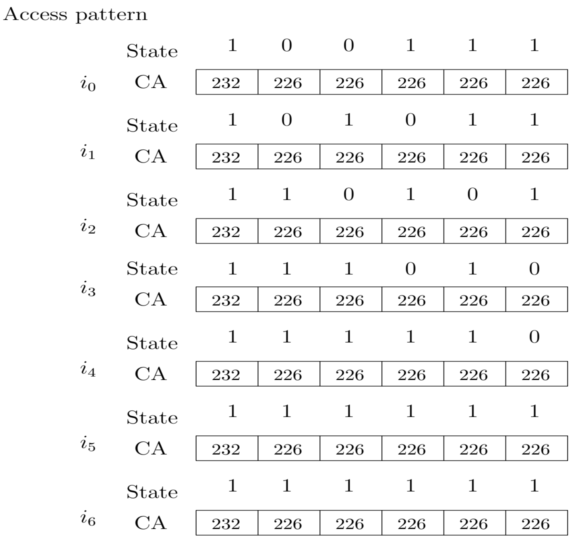

Fig. 12. Space time evolution of the first stage with hybrid rule < 232 226 226 226 226 226 >

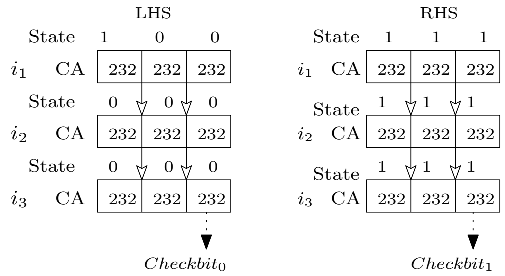

Fig. 13. Space time evolution of final stage

address in consultation with the memory controller.

Let S be a non-empty group with a defined operation '*' in it. T be a subgroup of S then { a*T | a $\in$ S } is called a coset. In this case, it is called left coset whereas T*a is called right coset. For normal subgroup, a*T=T*a. Cosets always partition the set into disjoint sets. For normal subgroup, cosets are identical or disjoint sets. The number of cosets is defined by the Equation 3 [32].

$$[S:T] = \frac{S_k}{T_k} \tag{3}$$

Group of all memory words (S) is $< Z_{256}, + >$ and $< Z_{16}, + >$ represents a subgroup (T). Therefore number of the cosets is 16 as per Equation 3. Therefore, the number of blocks per set is 16 (N) with 16 number of sets (M). Here operation * is represented by the Equation 4.

$$a * b = (a+b) mod(n-tuples) \ldots \{a, b \in S\} \tag{4}$$

The set of distinct cosets or partitions (P) represents the quotient group and given by Equation 5.

$$Q = \{\{0+H\}, \{1+H\}, \ldots, \{15+H\}\} \tag{5}$$

From the physical address, we can find out the bank and cache block address. Block address can be as per Equation 6.

$$MB_i = M * n + j \tag{6}$$

Here i denotes the block index. Set index (j) and block offset (n) variables are the remapping parameters. These can be varied from 0 to M-1 and 0 to N-1 respectively to adjust remapping offsets. To change within set or local wear leveling, block offset (n) and for global wear leveling set index (j) has to be varied in Equation 6. Let us take an example here with length of memory is 1024. S = $< Z_{1024}$,+ $>$ represents group of all memory words and T = $< Z_{16}$,+ $>$ represents subgroup of memory block within a set. Distinct cosets/partitions are collection of blocks with index numbers are P_0 = <0, 16, 32, 48, 64..>, P_1 = <1, 17, 33, 49, 65..>, ..P_{15} = <15, 31, 47, 63..>. Block of index number 49 can be written as MB_{49}=16*3 + 1. For global (interset) mapping the set index is changed from 1 to 6. So, the new block address will be 54 belonging to set 6. For local mapping j is to be varied in Equation 6.

VII. Performance Analysis

The *density classification task (DCT)* is a standard tool incorporated in the design to assess the *spatial locality of reference* or to identify the reused set of contiguous memory blocks. Write pressure is targeted to be reduced on those spatially located blocks using redirection of writes to another contiguous less written blocks. Source or target remapping zones are distinguished by the attractors of the employed *CA*s in consecutive two-stage operations as described in Section V. The Access pattern array is saved with a nonzero value to the corresponding position of the block for indicating write operation on it. In Table VI, all the possible number of states are classified as per the presence of number of 1s within the states. Those states, having less than 50% of 1s in their access pattern, fall on 0-basin and having more than 50% of 1s, fall on 1-basin for CA < 232 226 226 226 226 226> as shown in Table VI and for CA <232 184 184 184 184 184> as shown in Table Table VII. Almost uniformly distributed 0s and 1s states fall in α-basin (Table VI and VII). Few states are falling on opposite basin introducing errors in the design.

Hybridized CA with rule 232 & 184 and rule 232 & 226 can perform the same function in the following design and can be employed as alternate rules in the first stage. But they will show a relative advantage and/or disadvantage in performance in terms of misprediction rate and speed of execution. PBCA with rule 232 and rule 226, have a single state '48' (110000) associated with 1-basin causes mispredication. State '30' (011110) which falls in 0-basin, is another mispredicted state that introduces error in the design. Though few states with equal number of 0s (1s) in their bit

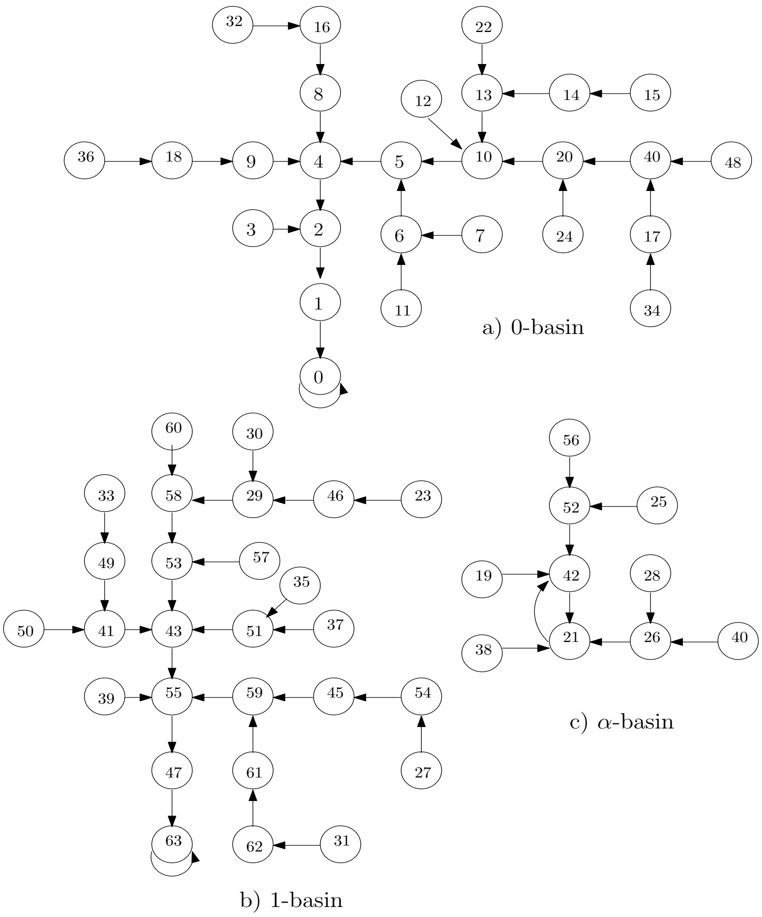

Fig. 14. STD for 6-cell < 232 184 184 184 184 184 > hybrid PBCA

TABLE VI. DESIGN ANALYSIS REPORT WITH CA< 232 226 226 226 226 226 >

Number of ones	1	2	3	4	5
States	1, 2, 4, 8, 16, 32	3, 5, 6, 9, 10, 12, 17, 18, 20, 24, 33, 34, 36, 40, 48	7, 11, 13, 14, 19, 21, 22, 25, 26, 28, 35, 37, 38, 41, 42, 44, 52, 56	15, 23, 27, 29, 30, 39, 43, 45, 46, 49, 50, 53, 54, 57, 58, 60	31, 47, 51, 55, 59, 61, 62
basin-0	1, 2, 4, 8, 16, 32	3, 5, 6, 9, 10, 12, 17, 18, 20, 24, 33, 34, 36, 40	13, 14, 22, 26, 28	30	-
basin-1	-	48	41, 52, 56	15, 23, 29, 39, 43, 45, 46, 49, 50, 53, 54, 57, 58, 60	31, 47, 51, 55, 59, 61, 62
basin-α	-	-	7, 11, 19, 25, 27 35, 37, 38, 42, 44	-	-
Misprediction	-	48 has fallen in basin-63	-	30 has fallen in basin-0	

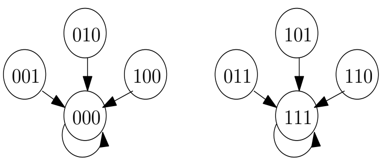

Fig. 15. State transition diagram of 3-cell uniform PBCA<232 232 232>

pattern fall on 1-basin (3 states) or 0-basin (5 states) does not cause any misprediction to insert any errors. Rule 232 and 184 have two mis-predicted states. State '15' fall in 0-basin (having greater than 50% of 1s) and '33' fall in 1-basin (having lesser than 50% of 1s) will contribute a little bit error in the design as they fall in opposite basin. Here also few states (5 states) with equal distribution fall on 0-basin or 1-basin (3 states) but doesn't contribute to any error contents as above.

It is observed that both the rules used for first stage have an α basin present on the state transition diagams (STDs) of CA $<$ 232 226 226 226 226 226 $>$ and CA $<$ 232 184 184 184 184 184 $>$ as shown in Figure 9 and Figure 14. As per requirement 4 ($Req4$), apart from all 0s and all 1s attractor no other attractors are permissible in proposed selection logic. Though no single length attractor is found, but one multilength attractor ($21 \rightarrow 42 \rightarrow 21$) is found in STDs. Hence the presence of this multilength loop ($21 \rightarrow 42 \rightarrow 21$) in them is violating the requirement ($Req4$) of the cellular automata (CA) based remapping candidate selection logic. But this basin is associated with all the uniformly distributed states for both the rules. Therefore, the presence of 'α' basin does not contribute any error in the design. Recollecting that this zone is defined as neither write stressed nor seldom written. Comparative analysis of the PBCA with

TABLE VII. DESIGN ANALYSIS REPORT WITH CA< 232 184 184 184 184 184 >

Number of ones	1	2	3	4	5
States	1, 2, 4, 8, 16, 32	3, 5, 6, 9, 10, 12, 17, 18, 20, 24, 33, 34, 36, 40, 48	7, 11, 13, 14, 19, 21, 22, 25, 26, 28, 35, 37, 38, 41, 42, 44, 52, 56	15, 23, 27, 29, 30, 39, 43, 45, 46, 49, 50, 53, 54, 57, 58, 60	31, 47, 51, 55, 59, 61, 62
basin-0	1, 2, 4, 8, 16, 32	3, 5, 6, 9, 10, 12, 17, 18, 20, 24, 34, 36, 40, 48	7, 11, 13, 14, 22	15	-
basin-1	-	33	35, 37, 41	23, 27, 29, 30, 39, 43, 45, 46, 49, 50, 53, 54, 57, 58, 60	31, 47, 51, 55, 59, 61, 62
basin-α	-	-	19, 25, 26 28, 38, 42, 44, 52, 56	-	-
Misprediction	-	33 has fallen in basin-63	-	15 has fallen in basin-0	-

TABLE VIII. COMPARATIVE STATE ANALYSIS REPORT

Types of Basin	basin-0		basin-1		basin α	
Rules	232 & 184	232 & 226	232 & 184	232 & 226	232 & 184	232 & 226
States	0, 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	0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 13, 14, 16, 17, 18, 20, 22, 24, 26, 28, 30, 32, 33, 34,36, 40, 48	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	7, 11, 15, 23, 27, 29, 31, 38, 39, 43, 45, 46, 47, 51, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63	19, 25, 26, 28, 38, 42, 44, 52, 56	19, 25, 35, 37, 41, 42, 44, 50, 52
Cycles	One single length loop 0→0	One single length loop 0→ 0	One single length loop 63→63	One single length loop 63→63	21 → 42 → 21	21 → 42 → 21
Misprediction	15(001111) as it falls on '0' basin having greater number of 1s	30(011110) as it falls on '0' basin having greater number of 1s	33(100001) as it falls on '1' basin having less number of 1s	48 (110000) as it falls on '1' basin having less number of 1s	No unpredictable states	No unpredictable states
Depth of CA	9	11	9	11	9	11

rule 232 and 184 and PBCA with rule 232 and 226, are given in Table VIII. The rules are compared in terms of number of single length cycles (for finding attractors) to establish the design requirements ($Req1$ and $Req2$). Further, the depth of CA is found to get the maximum number of required clock cycles for execution of first stage. It is found that depth of CA (11) is greater for PBCA < 232 226 226 226 226 226> than PBCA < 232 184 184 184 184 184 > (9). Therefore, previous one requires more clock cycles to execute as compared to the PBCA with rule 232 and rule 184 as per Table VIII.

Efficiency of wear leveling scheme is a function of size of the remapping zone [8] which is calculated in terms of number of contiguous memory blocks of remapping zone. Here, the size is three taking central block, left and right neighbouring block. The final stage is a 3-cell uniform CA which decides the remapping candidates. To keep minimum size of the zone, only three memory blocks are considered from the write dominated zone which is identified in final stage selected from the first stage. Final stage is executed in a single clock cycle as shown in Figure 15 of STD of 3-cell Uniform PBCA < 232 232 232>. But using two-stage hierarchical operation may increase design complexity of memory controller in terms of computation and space overhead.

Remapping method can be well explained with the help of Table IX. RL and RG are defined as local and global remapping methods respectively to fulfill the requirement of wear leveling in ReRAM. Three different sized memory 256 B, 512 B and 1024 B are considered here with 16, 32 and 16 number of sets. Remapping parameters are block-offset (TB_0) and set-index (TS_i) which are defined over local (RL) and global (RG) methods. Source set index (SS_i) and source block offset (SB_0) are varied accordingly to find the target block address for different source blocks. Algebraic remapping method is therefore accomplished in between intraset and interset remapping to reduce write-stress on memory blocks to improve the durability of memory cells.

VIII. CONCLUSION AND FUTURE WORK

In this paper, we have proposed a design frame work for a typical wear leveling scheme which is intended to apply on resistive memory using periodic boundary one dimentional two states cellular automata (CA). Write distribution on memory blocks are tried to be kept uniform by remapping of intended write request within shared LLC of chip multiprocessors (CMPs). This scheme can improve cell lifetime by local/global wear leveling techniques though it incurs a hardware overhead of the design. This work addresses wear leveling due to repetitive writes in resistive memory caused by the spatial locality of reference. But write access pattern not only shows spatial locality of access, it also shows the spatial correlation among the accessed blocks. It is due to the reason of data storage pattern (data structure). Therefore, the wear leveling can also be considered for spatially correlated memory blocks by observing the data storage patterns.

Here, in our proposed wear leveling scheme for resistive memory, uneven or non-uniformity of writes due to workload distribution of chip multiprocessors are considered as the reason of repetitive writes in memory. It is not addressing write-stress caused due to malicious attacks. Therefore, this periodic boundary cellular automata based technique can be

TABLE IX. TABULAR REPRESENTATION OF REMAPPING METHOD

Type		Memory			Remapping parameter			Source			Target			Target address		
RL	RG	M_{256} & S_{16}	M_{512} & S_{32}	M_{1024} & S_{16}	n_{256} or j_{256}	n_{512} or j_{512}	n_{1024} or j_{1024}	SS_i & SB_o	SS_i & SB_o	SS_i & SB_o	TS_i & TB_o	TS_i & TB_o	TS_i & TB_o	T_{256}	T_{512}	T_{1024}
$\surd$	-	17	39	97	$1 \rightarrow 5$	$7 \rightarrow 6$	$6 \rightarrow 7$	1,1	7,1	1,6	5,1	6,1	5,6	21	199	113
-	$\surd$	17	39	97	$1 \rightarrow 5$	$1 \rightarrow 6$	$6 \rightarrow 7$	1,1	7,1	1,6	1,5	7,6	1,7	81	38	101
$\surd$	-	65	69	101	$4 \rightarrow 6$	$2 \rightarrow 6$	$6 \rightarrow 8$	1,4	5,2	5,6	1,6	5,6	5,8	97	197	133
-	$\surd$	65	69	101	$1 \rightarrow 6$	$5 \rightarrow 6$	$5 \rightarrow 10$	6,4	5,2	5,6	1,6	5,6	10,6	70	70	106
$\surd$	-	215	508	1021	$13 \rightarrow 5$	$15 \rightarrow 6$	$16 \rightarrow 9$	7,13	28,15	13,16	7,5	5,6	13,9	87	220	580
-	$\surd$	215	508	1021	$7 \rightarrow 2$	$28 \rightarrow 19$	$13 \rightarrow 7$	7,13	28,15	13,16	2,13	19,15	7,16	210	499	1015
$\surd$	-	107	436	735	$6 \rightarrow 1$	$13 \rightarrow 3$	$45 \rightarrow 15$	11,6	20,13	15,45	11,1	20,3	15,15	27	116	255
-	$\surd$	153	364	735	$9 \rightarrow 2$	$12 \rightarrow 4$	$15 \rightarrow 9$	9,9	12,11	15,45	2,9	4,11	9,45	41	356	729

extended for prevention of malicious attacks by capturing temporal as well spatial access patterns.

REFERENCES

[1] K. Asanovic, J. Beck, B. Irissou, B. E. D. Kingsbury and J. Wawrzynek, "T0: A Single-Chip Vector Microprocessor with Reconfigurable Pipelines," *ESSCIRC '96: Proceedings of the 22nd European Solid-State Circuits Conference*, Neuchatel, Switzerland, 1996, pp. 344-347.

[2] N. N. Sirhan and I. S. Serhan, "Multi-core Processors: Concepts and Implementations," *International Journal of Computer Science and Information Technology*, vol. 10, no. 1, pp. 1-10, 2018.

[3] T. Ao, P. Chen, Z. He, K. Dai and X. Zou, "RDCC: A New Metric for Processor Workload Characteristics Evaluation," *IAENG International Journal of Computer Science*, vol. 40, no. 4, pp. 274-284, 2013.

[4] T. Zhang, N. Wu, F. Zhou, L. Zhou, and X. Zhang,"A Traffic Equilibrium Mapping Method with Energy Minimization for 3D NoC-Bus Mesh Architecture," *IAENG International Journal of Computer Science*, vol. 42, no. 1, pp. 1-7, 2015.

[5] N. P. Khanyile, J. R. Tapamo, and E. Dube, "An Analytic Model for Predicting the Performance of Distributed Applications on Multicore Clusters," *IAENG International Journal of Computer Science*, vol. 39, no. 3, pp. 312-320, 2012.

[6] 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*, vol. 23, no. 10, pp. 2149-2161, 2015.

[7] S. Mittal, Y. Cao and Z. Zhang, "MASTER: A Multicore Cache Energy-Saving Technique Using Dynamic Cache Reconfiguration," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 8, pp. 1653-1665, 2014.

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

[9] 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," *2009 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, New York, NY, 2009, pp. 14-23.

[10] L. P. Chang, "On Efficient Wear Leveling for Large-scale Flash-memory Storage Systems," *SAC '07: Proceedings of the 2007 ACM symposium on Applied computing*, March 2007, pp. 1126-1130.

[11] L. Zhu, Z. Chen, F. Liu and N. Xiao, "Wear Leveling for Non-Volatile Memory: a Runtime System Approach," *IEEE Access*, vol. 6, pp. 60622-60634, 2018.

[12] M. K. Qureshi, A. Seznec, L. A. Lastras and M. M. Franceschini, "Practical and secure PCM systems by online detection of malicious write streams," *2011 IEEE 17th International Symposium on High Performance Computer Architecture*, San Antonio, TX, 2011, pp. 478-489.

[13] G. Wu, H. Zhang, Y. Dong and J. Hu, "CAR: Securing PCM Main Memory System with Cache Address Remapping," *2012 IEEE 18th International Conference on Parallel and Distributed Systems*, Singapore, 2012, pp. 628-635.

[14] W. Wen, Y. Zhang and J. Yang, "Wear Leveling for Crossbar Resistive Memory," *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, San Francisco, CA, 2018, pp. 1-6

[15] S. Mittal and J. S. Vetter, "EqualWrites: Reducing Intra-Set Write Variations for Enhancing Lifetime of Non-Volatile Caches," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 1, pp. 103-114, 2016.

[16] M. Dalui and B. K. Sikdar, "A Cellular Automata Based Self-correcting Protocol Processor for Scalable Cmps," *Microelectronics Journal* vol. 62, pp. 108-119, 2017.

[17] S. Sarkar, M. Saha and B. K. Sikdar, "Multi-bit fault tolerant design for resistive memories through dynamic partitioning," *2017 IEEE East-West Design Test Symposium (EWDTS)*, Novi Sad, 2017, pp. 1-6.

[18] B. Das, S. Kamilya and B. K. Sikdar, "Design of CA based scheme for evenhanded data migration in CMPs," *2016 Sixth International Symposium on Embedded Computing and System Design (ISED)*, Patna, 2016, pp. 117-121.

[19] Mousumi Saha, and Biplab K Sikdar, "Test Structure for L1 Cache in Tiled CMPs," IAENG International Journal of Computer Science, vol. 43, no.3, pp392-401, 2016

[20] M. Saha, B. Das and B. K. Sikdar, "Periodic boundary cellular automata based test structure for memory," *2017 IEEE East-West Design Test Symposium (EWDTS)*, Novi Sad, 2017, pp. 1-6.

[21] A. Jadidi, M. Arjomand and H. Sarbazi-Azad, "High-endurance and performance-efficient design of hybrid cache architectures through adaptive line replacement," *IEEE/ACM International Symposium on Low Power Electronics and Design*, Fukuoka, 2011, pp. 79-84.

[22] J. Li, C. J. Xue and Yinlong Xu, "STT-RAM based energy-efficiency hybrid cache for CMPs," *2011 IEEE/IFIP 19th International Conference on VLSI and System-on-Chip*, Hong Kong, 2011, pp. 31-36.

[23] M. A. V. Rodriguez, J. M. S. Perez, J. A. G. Pulido, "An educational tool for testing caches on symmetric multiprocessors," *Microprocessors and Microsystems,*, vol. 25, no. 4, pp. 187-194, 2001.

[24] 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 Design Automation Conference*, Sydney, NSW, 2012, pp. 317-322.

[25] C. Stone and L. Bull, "Solving the Density Classification Task Using Cellular Automaton 184 with Memory," *Complex Systems*, vol. 18, no. 3, 2009.

[26] H. Fuks, "Solution of the Density Classification Problem with Two Cellular Automata Rules," *American PHYSICAL REVIEW E*, vol. 55, no. 3, pp. R2081-R2084, 1997.

[27] J. Hu, M. Xie, C. Pan, C. J. Xue, Q. Zhuge and E. H-M. Sha, "Low Overhead Software Wear Leveling for Hybrid PCM + DRAM Main Memory on Embedded Systems," *in IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 4, pp. 654-663, 2015.

[28] C. Wang and W. Wong, "Observational wear leveling: An efficient algorithm for flash memory management," *DAC Design Automation Conference 2012*, San Francisco, CA, 2012, pp. 235-242.

[29] C. Y. 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.

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

[31] J. V. Neumann, "The Theory of Self-reproducing Automata," *In Tech Report 03105-30-T, ORA, University of Michigan*, 1964.

[32] J. Nicholoson, "The Development and Understanding of the Concept of Quotient Group", *HISTORIA MATHEMATICA*, vol. 20, pp. 68-88, 1993.

[33] A. Seznec, "A Phase Change Memory as a Secure Main Memory," *IEEE Computer Architecture Letters*, vol. 9, no. 1, pp. 5-8, 2010.

Sutapa Sarkar received her M.Tech degree from Indian Institute of Engineering Science and Technology, Shibpur in 2008. Presently, she is working as Head of the Department (Assistant Professor) in Seacom Engineering College in the department of Electronics & Communication Engineering. She is also a Corporate Member of Institution of Engineers (India). Now, she is pursuing her PhD in CST department of IIEST-Shibpur. She is having research interests in Computer Architecture, Digital Signal processing and recent research trends in VLSI and on the development of Cellular Automata theory and its applications.

Manisha Ghosh received the B.E. degree from the University Institute of Technology, Burdwan University in 2008. She received the M.Tech degree from the University of Burdwan in 2010. Presently, she is working as Assistant Professor in the Department of ECE in NSHM Knowledge Campus, Durgapur. She is associate member of Institution of Engineers (India). She has research interest in Switching Theory, Digital Circuit, Computer Architecture, VLSI Design and Tesing, and Cellular Automata.

Biplab K Sikdar Received the B. Sc. (Hons) degree in Physics from Presidency College, Calcutta University, in 1985 and B. Tech and M. Tech degrees in Computer Science and Engineering from Calcutta University, India, in 1988 and 1990, respectively, and the PhD degree in Engineering from Bengal Engineering College (a Deemed University), Howrah, India in 2003. He was the faculty of Computer Science and Engineering in North Eastern Regional Institute of Science and Technology, India from 1991 to 1992 and in University of North Bengal, India, from 1992 to 1997. Presently he is a professor in the Department of Computer Science and Technology, IIEST, Shibpur, West Bengal, India. His research interests include digital system design and test. He has been working on the development of Cellular Automata theory and its applications.

Mousumi Saha received the B.E. degree in Computer Science and Engineering from the Regional Engineering College (Now National Institute of Technology), Durgapur, West Bengal, India in 1997 and M.Tech degree in CSE from Calcutta University, West Bengal in 2001. She received PhD from the CSE department, IIEST, Shibpur, West Bengal. She is currently working at National Institute of Technology, Durgapur as an Assistant professor in the department of CSE. Her research interests include VLSI Design and Testing, Theory of Cellular Automata and its application, Visual Light Communication, Societal Computing and Internet of Things.