MEMRES: A Fast Memory System Reliability Simulator

Shaodi Wang, *Student Member, IEEE*, Henry (Chaohong) Hu, Hongzhong Zheng, *Member, IEEE*, and Puneet Gupta, *Member, IEEE*

***Abstract*—With scaling technology, emerging nonvolatile devices, and data-intensive applications, memory faults have become a major reliability concern for computing systems. With various hardware and software approaches proposed to address this issue, a comprehensive evaluation is required to understand the effectiveness of these solutions. Considering the complex nature of various memory faults as well as interactions between various correction mechanisms, we propose MEMRES, a fast main memory system reliability simulator. It enables memory fault simulation with error-correcting code (ECC) algorithms and modern memory reliability management, including memory page retirement, mirroring, scrubbing, and hardware sparing. MEMRES is computationally efficient in obtaining memory failure probabilities in the presence of multiple failure mechanisms and complex correction scheme, allowing the optimization of memory system reliability, the prediction of emerging memory reliability, and designing a reliability enhancement technique. The accuracy of MEMRES is verified by an existing analytical model and an existing memory fault simulator. We performed a case study on spin-transfer torque random access memory (STT-RAM)-based main memory, and the results indicate that in-memory ECC can significantly mitigate the write error rate of STT-RAM, demonstrating the capability of handling emerging memory system.**

***Index Terms*—Memory fault, memory mirroring, memory page retirement, memory reliability, magnetic random access memory (MRAM), reliability management, retention error, simulator, sparing, spin-transfer torque random access memory (STT-RAM), write error.**

NOMENCLATURE

λ	Failure rate.
BL	Burst length.
k	Number of faulty symbols.
M	Memory size.
$MapSsize$	Number of addresses in an FM/AM.
N_S	Number of symbols in an ECC word.
N_{MFS}	Maximum correctable symbols of an in-controller ECC.
N_{PFB}	Number of possible faulty bits in a symbol.
$P_{\mathrm{correct\_symbol}}$	Probability that a symbol covered by an FM is ECC correctable.
P_{FB}	Probability that a bit covered by an FM is faulty.
$P_{\mathrm{wrErr\_fail\_ctrECC}}$	Probability that an in-controller ECC failure is caused by the intersection of write/retention error and memory faults.
$P_{\mathrm{wrErr\_fail\_memECC}}$	Probability that an occurrence of a write/retention error in an accessed symbol causes an in-memory SECDED failure.
SL	Symbol length (number of bits output from a chip in an access).
$P_{\mathrm{correct\_word}}$	Probability that a word covered by an FM is ECC correctable.
P_{fail}	Probability that memory failure is caused by an intersection of one memory fault and one transient fault.
$P_{\mathrm{intersect}}$	Probability that two injected faults intersect.
$P_{\mathrm{perm}}\ (t_1)$	Probability that a permanent fault occurs at time t_1.
$P_{\mathrm{perm}}\ (t_2)$	Probability that a transient fault occurs at t_2.
AM	Access-map, a data structure that statistically models memory read/write operations on a memory space in a period of time.
Chipkill	ECC codes that can correct any errors from single chip.
Cover-Rate	Percentage of valid addresses in an FM/AM.
DDDC	Double-device data correction.
DIMM	Dual in-line memory module.
DRAM	Dynamic random access memory.
ECC	Error-correcting code.
FIT	Expected number of failures in one billion device-hour.
FM	Fault-map, the basic data structure in MEMRES, describing the address, coverage, and number of faulty addresses in a faulty memory space.
I	Intersection of A and B.
SBT	Single-bit transient errors, including data-link error, retention error of STT-RAM and write error of STT-RAM.
SBTER	Single-bit transient error rate.

Manuscript received December 17, 2015; revised April 25, 2016; accepted August 24, 2016. This work was supported by the NSF Variability Expedition under Grant CCF-1029783. Associate Editor: E. Pohl.

S. Wang and P. Gupta are with the Department of Electrical Engineering, University of California, Los Angeles, CA 90095 USA (e-mail: shaodiwang@g.ucla.edu; puneet@ee.ucla.edu).

H. (C.) Hu and H. Zheng are with Samsung Semiconductor, San Jose, CA 95134 USA (e-mail: henry.hu@ssi.samsung.com; hz.zheng@ssi.samsung.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TR.2016.2608357

SCCDCD	Single-chip-correction-double-chip-detection.
SECDED	Single-error-correction-double-error-detection.
STT-RAM	Spin-transfer torque random access memory.

I. Introduction

RECENTLY, many datacenter studies [1]–[4] point out that main memory reliability is becoming a crucial problem in computing systems. Although the per-bit memory failure rate improves with technology development, the improvement cannot compensate the growing memory density required by increasing data-heavy applications, and hence worsened memory failure rate has been observed [4]. Moreover, the possible introduction of emerging nonvolatile memories in the future will exacerbate the reliability problem [5], [6]. The consequences of memory failure are frequent system failure recovery and faulty memory replacement, which result in reduced serviceability and increased costs. Improving the reliability of a future memory system requires a good understanding of memory failure mechanisms and reliability enhancement techniques.

Evaluation of modern memory systems is nontrivial, because memory failure is caused by a large variety of memory fault types [1], [2], [4], and various sophisticated techniques (see Fig. 1) have been proposed to repair the faults. Most previous evaluation studies have relied on analytical models, e.g., [7]. As is illustrated in Fig. 1, the models are insufficient to handle a variety of memory fault types simultaneously, to capture the effects of reliability enhancement techniques and to incorporate application influence. First, memory failure is caused by multiple fault types and their interaction; however, developing models that include all fault types and interactions is an impractical task, and missing a fault type may result in significant accuracy loss. For example, the data-link error [8], [9] and nonpersistent error of emerging memories [10], [11] are not considered in any existing models, and their interaction with other memory faults can induce the most failures that crash memory systems. Second, a memory failure model is strongly dependent on reliability enhancement techniques; sophisticated ECC, e.g., DDDC [12], [13], and memory reliability management (see Fig. 1) dramatically add to the modeling difficulty. Third, fault rate varies with application and time [3], [4], but analytical models assume a constant fault rate.

Experimentally analyzing memory faults requires a large statistical experiment setup. Field studies [1]–[4] have recorded and analyzed memory errors in data centers for over a year. Despite the high cost, conflicting conclusions exist in these studies for lacking the access to finer granularity of memory fault interaction and uncontrolled hardware design variables. In addition, due to the dependence of memory fault on application and memory architecture [3], the conclusion from a field study is difficulty to use to predict the reliability of other computing systems. Therefore, efficient and flexible simulation methodologies that can perform finer analyses are required in memory reliability study.

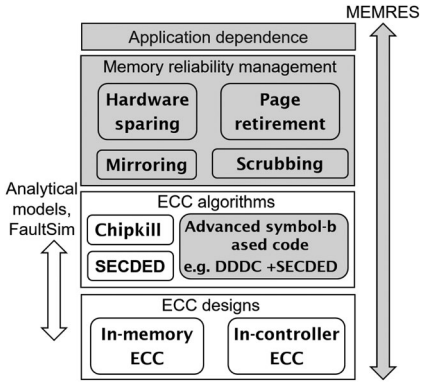


Fig. 1. Limitation of analytical models and FaultSim on memory reliability evaluation. The reliability enhancement techniques in gray boxes can only be evaluated by MEMRES.

Various memory fault simulators have been developed and used in industry and academia [14], [15]. However, these simulators are unsuitable for the purpose of obtaining memory failure probability. They focus on fault propagation, which involves great computation of logic circuit topology simulation [16]. To obtain failure probability requires many Monte Carlo simulations, which lead to unacceptably run time (e.g., million device-years). FaultSim is a recent developed high-speed Monte Carlo DRAM fault simulator [17]. It takes a few hours (seconds in the event mode) to obtain years-long DRAM failure probability. However, FaultSim does not support modern memory system fault simulation due to several missed models, e.g., memory access behavior and memory reliability management.

An effective memory reliability simulator should be able to handle the advanced techniques used in state-of-the-art memory systems while minimizing the computation involved in the simulation. We propose MEMRES, an efficient memory reliability simulator that satisfies this need. It performs long-term (i.e., month-year) memory system reliability simulation. Table I compares the analytical model [7], FaultSim [17], and MEMRES. In addition to supporting all features of the analytical model and FaultSim, MEMRES enables simulation with varying fault rate/density and memory access density/distribution, in-memory and in-controller ECCs, and modern reliability enhancement techniques. As examples, MEMRES differently models fault with faulty bit density according to the truth that a fault usually has $< 1\%$ faulty bits in its coverage (e.g., a bank fault only affects $< 1\%$ rows) [1], [4], and MEMRES adds the memory access into fault simulation, where a fault is only activated after being accessed. To support finer granularity of memory system fault simulation without involving large run time overhead, we developed statistical models for most of new features in MEMRES, which are described in the following sections. they enable MEMRES to run as fast as FaultSim's interval mode (see run time in Table I). Due to the finer memory modeling and new features, MEMRES would not be sped up like FaultSim by the event mode. However, MEMRES supports parallel computation, which takes 50 min to complete 100 000 5-year of memory reliability Monte Carlo simulations using eight threads.

TABLE I
COMPARISON WITH EXISTING MEMORY FAULT ANALYSIS METHODS

		Analytical model [7]	FaultSim [17]	MEMRES
ECC	SECDED, Chipkill	✓	✓	✓
	Advanced ECCs, e.g., V-ECC [18], SWD-ECC [19]	×	✓	✓
	In-controller ECC	✓	✓	✓
	In-memory ECC	×	×	✓
Fault model (FIT)	Constant FIT	✓	✓	✓
	Temporal variation (Fig. 16)	×	×	✓
	Spacial variation	×	×	✓
	Data-link and retention errors	×	TSV errors	✓
Memory access model	Uniform access	✓	✓	✓
	Temporal variation	×	×	✓
	Spacial variation (Fig. 12)	×	×	✓
Memory reliability management	Scrubbing	×	✓	✓
	Hardware sparing (Fig. 13)	×	×	✓
	Page retirement (Fig. 14)	×	×	✓
	Mirroring (Fig. 15)	×	×	✓
Run time		< 1 ms	6.6 h (interval) 6.7 s (event)	5.2 h (1 thread) 50 min (8 threads)

Run time is measured using single-core on AMD Opteron Processor 2380. FaultSim has event mode, which uses analytical models to partially replace Monte Carlo fault injection in regular interval mode to speedup simulations.

Compared with existing silicon-based memories, emerging memories potentially suffer more severe reliability problems. MEMRES is capable of predicting the reliability of emerging memories. To demonstrate this capability, we use STT-RAM [20]–[22] as a vehicle to explore optimized designs of different memory reliability enhancement techniques. STT-RAM promises the speed, area, and endurance of DRAM [23] while holding the nonvolatility, and hence, is identified as a possible replacement of DRAM [24]. However, STT-RAM faces the challenge of write errors and retention errors due to process variation [11], [25]–[29]. Adding models of the new memory errors are convenient in MEMRES.

Our contributions are summarized as follows:

1) We implement an fast memory reliability simulator. This simulator takes minutes to obtain precise memory failure probability and reasons.
2) MEMRES can model memory access behavior, which differs from applications. The access behavior was ignored by existing models and reliability simulators, but has been demonstrated to significantly impact fault characteristics.
3) To our best knowledge, memory reliability management is for the first time modeled and simulated including memory scrubbing, row/column/rank sparing, memory page retirement, and memory mirroring.
4) Multiple failure mechanisms including data-link errors [8], [9] and nonpersistent errors [10] are modeled and demonstrated to affect the effectiveness of reliability enhancement techniques. These transient errors are ignored by previous studies.
5) New models are derived including the failure probability model of memory system with both in-memory ECC and in-controller ECC (see Fig. 7) and the interaction between transient faults and permanent faults.
6) We improved the data structure used in [17] by adding the parameters Cover-Rate and Access-Rate, which improve simulation accuracy.
7) The reliability of an STT-RAM based main memory and effectiveness of memory reliability enhancement techniques (ECC designs, algorithms, and memory reliability techniques) are evaluated.

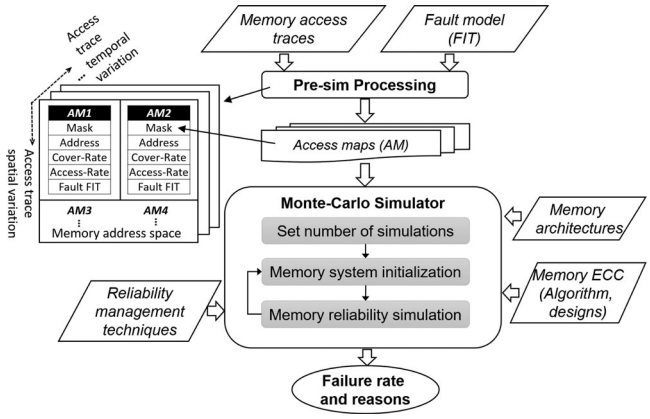


Fig. 2. Framework overview of MEMRES.

The paper is organized as follows. Section II describes data structures, basic operations, and models used in MEMRES. Section III validates MEMRES by the analytical model [7], derived models for transient faults, and FaultSim [17]. Section IV evaluates the reliability of STT-RAM designs with different retention time, write time, ECC designs, and algorithms, memory reliability managements, and fault models. Section V concludes our work.

II. MEMRES SOFTWARE FRAMEWORK

Fig. 2 illustrates the overview of MEMRES tool flow. MEMRES comprises of two components: pre-sim processing and the Monte Carlo simulator.

A. Presim Processing

The presim processing is a one-time procedure for modeling memory fault and memory access behavior. Simply incorporating memory traces in simulation is impractical due to the fact that the memory traces occupy large memory space (one-second memory traces require several GB storage space) and dramatically affects simulation speed. In MEMRES, pre-sim processing extracts memory access density distribution from memory access traces and passes it to the Monte Carlo simulator in the form of AM, which is a basic data structure in MEMRES. An AM models the access density on a specified memory address space (e.g., one column, one bank, one channel, etc.) in a specified

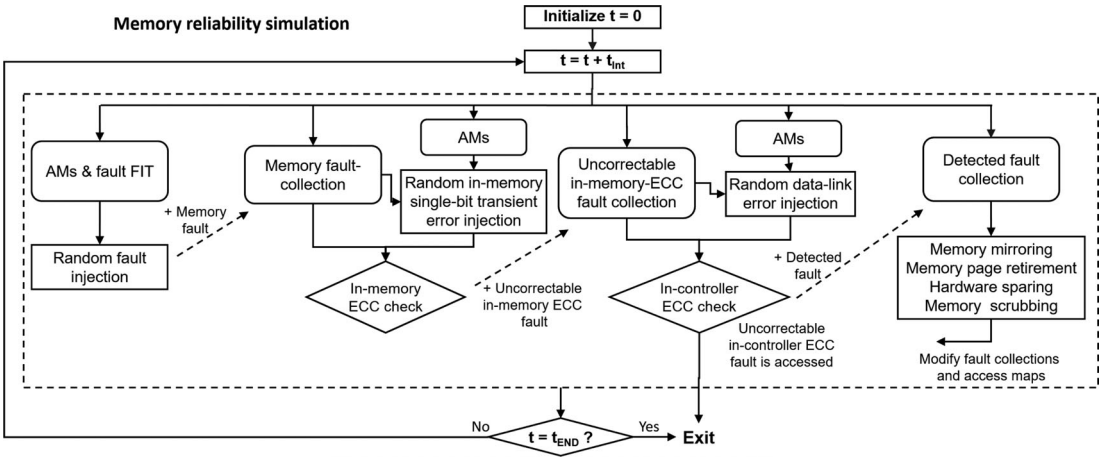


Fig. 3. Memory reliability simulation. The simulation divides years-long memory lifetime into short intervals. In each interval, events of random fault injection, ECC checks, and memory reliability management are simulated. The simulation terminates when an uncorrectable fault occurs or it reaches the end of preset simulation time.

period. As is shown in Fig. 2, memory access spacial variation in a time period (e.g., some space is more intensively accessed) can be captured by multiple AMs, and temporal variation (e.g., memory access density differs from time) can also be described by having more AMs. Hence, access behavior of one application can be captured by a set of AMs. In addition, a server running multiple applications can be described by passing multiple sets of AMs alternatively to simulator. The presim processing also calculates fault failure-in-time (FIT, expected number of failures in one billion device-time) for each AM individually according to the memory fault model (e.g., more frequently accessed AM has higher write error rate).

B. Monte Carlo Simulator

The Monte Carlo simulator performs a number of memory reliability simulations, and every simulation simulates a memory's lifetime (over years) reliability behavior. These simulations differ from each other due to random event modeling like fault injection, and hence the memory failure rate and reasons can be statistically extracted from them. The configuration of memory architecture, reliability designs including ECC designs, ECC algorithms, and memory reliability management is input of MEMRES. The procedure of one memory's lifetime simulation is shown in Fig. 3. The simulation divides memory lifetime into short intervals and then simulates one interval after another. In each interval, four events are simulated including random fault injection, in-memory ECC check, in-controller ECC check, and memory reliability management. In the module of random fault injection, the probability of fault occurrence in one interval is first calculated according to fault FIT rate. Then, faults are randomly injected based on the calculated probability, and injected faults are stored in a memory fault-collection. In-memory single-bit transient errors like retention errors and write errors are also injected, which may intersect with the faults in the memory fault-collection. The fault injection models are detailed in Section II-E1. In the module of in-memory ECC check, injected faults are checked whether they can be corrected by the in-memory ECC. The uncorrectable in-memory ECC faults are added to an uncorrectable in-memory ECC fault-collection. These faults may intersect with injected data-link errors to produce uncorrectable errors. The faults and data-link errors are checked together against in-controller ECC. Once an uncorrected in-controller ECC fault is accessed by an AM, a memory failure is produced and terminates current simulation. The detailed ECC model is discussed in Section II-E2. The detectable faults are added to a detected fault-collection. In the module of memory reliability management, repairing techniques can be triggered by detected faults to physically repair memory devices or to systematically block accessing to the detected faults.

C. Fault-Map and Access-Map

In the field studies [1], [2], detected faults are classified to several fault types, e.g., a row, a column, and a bank. A fault type defines a faulty region where multiple memory errors are produced. Examples of fault types are illustrated in Fig. 4. Larger fault types are likely caused by logic circuit failure. For example, a sense amplifier fault may lead to read errors in a column (column fault), or a particle hitting a bank decoder may cause errors in a bank (bank fault). Individually storing the affected bits in the simulator data structure requires a huge memory space. Such a large fault can be represented by a single data structure. This structure is comprised of Mask and Address. Mask specifies the fault size, and Address locates the fault in memory space. This data structure is efficient in terms of computation speed and memory consumption. However, this structure assumed that all addresses in a fault are faulty, which is incorrectly. Most fault types have only $< 1\%$ faulty addresses in their covered memory space [1], [2]. In order to accurately model the fault behavior, we add another statistical parameter, Cover-Rate (ranging from 0 to 1), to represent the percentage of faulty addresses in a fault. The Mask, Address, and Cover-Rate comprise a FM, which is the basic data structure in MEMRES to represent faults.

In order to catch application-dependent fault behavior and model system-level reliability enhancement techniques, MEMRES uses AM to model memory access behavior. It has five parameters, including Mask, Address, and Cover-Rate, which

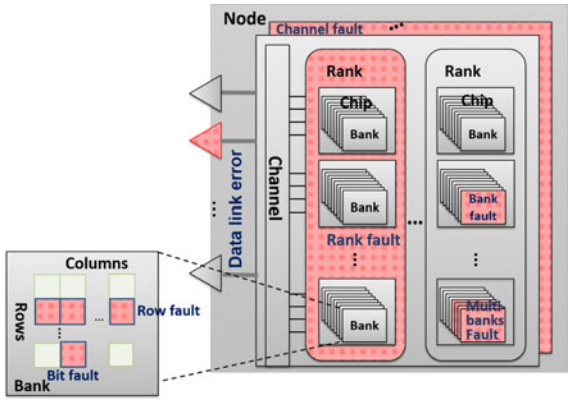
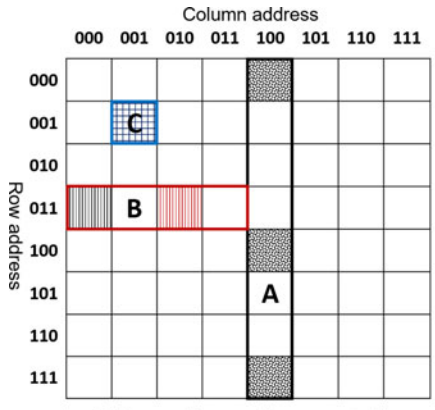


Fig. 4. Memory architecture and memory fault types. A bank is constructed by columns and rows (several rows are grouped in a mat, which is not shown in the figure). Eight banks are built in a chip (device), and nine × 8 chips (8 data chips + 1 ECC chip) or eighteen × 4 chips (16 data chips + 2 ECC chips) construct a rank. Several ranks are built in a channel. In a write or read operation, a channel and a rank is firstly selected by a decoder, and then a word is written/read across all chips in the selected rank, e.g., every × 8 chip in a rank outputs 8 bits to comprise a 72-bit word (64 data bits + 8 ECC bits), where all chips in a rank share the same addresses. A fault type is defined by the component that is affected, e.g., a bank fault indicates multiple faulty rows/columns in the bank.



Fault/Access Range Representation

	Mask		Address		Cover-Rate
	Row	Column	Row	Column	
A	111	000	000	100	0.375
B	000	011	011	000	0.5
C	000	000	001	001	1

Fig. 5. Examples of FM/AMs A, B, and C. A is a column FM/AM, B is a 4-bit FM/AM, and C is a single-bit FM/AM.

have the same definitions as in FM except that they model accessed memory space not faulty space. In addition, AM also contains another two parameters: Access-Rate representing access rate in an AM (number of read/writes in an hour) and FIT representing fault rate (expected number of faults in a billion device-hour). As is mentioned in Section II-A, memory access behavior changes over time and memory space creating the need of a set of AMs for different simulation intervals and memory space. Finer division in time and space leads to higher modeling accuracy, but results in more AMs, calculations, and simulation time. The tradeoff between speed and accuracy is analyzed in Section II-F.

Fig. 5 shows examples of FM/AMs A, B, and C. Address and Mask are sets of binary bits, which represent a region of device addresses (i.e., physical location in memory). In Fig. 5, FM/AM A only occupies the column 100, so its column address is 100. To tell MEMRES that A's three column address bits are valid (i.e., all these the three bits are required to determine A's location, so they are valid), A's column mask should be set to 000, where mask bit 0 means that the corresponding address bit is valid, and mask bit 1 means that the corresponding bit is invalid (masked). As A covers all rows from row address 000 to 111, row address bits are not necessary to determine A's position, its row mask is 111 to set all corresponding row address bits invalid. For calculation simplicity, address bit is set to 0 when it is invalid. Similarly, B locates in the row 011, hence, its row address is 011 and valid, and row mask is 000. As B covers columns 000, 001, 010, and 011, and only the first address bit determines B's position and is valid. Therefore, its column mask and address are 011 and 000, respectively, where the first address bit 0 indicates that B covers the left four columns. FM/AM C is a single bit, and all address bits are valid. The Cover-Rate is the percentage (or probability) of addresses being faulty in a fault. For instance, three out of eight addresses are faulty in A resulting in a Cover-Rate of 0.375.

The number of addresses covered by an FM must be a power of two. Most fault types can be represented by an FM, e.g., the number of addresses in all fault types shown in Fig. 4 are powers of 2. A fault with size S not exactly equaling a power of two can be divided into no more than $\log_2 S$ parts such that each part is represented by an FM. FM helps MEMRES to save memory space by orders of magnitude compared to traditional simulators.

D. Basic Operations

MEMRES's most operations are constructed by three basic bitwise operations: INTERSECT, MERGE, and REMOVE. They use FM/AM and have the closure property (i.e., both the operands and results of these operations are FM/AMs).

INTERSECT calculates the intersection between two FM/AMs. For examples, when a fault (represented by an FM) is accessed by an AM, the intersection between the FM and AM exists, and when two faults from different chips are accessed simultaneously in a word, their intersection exists. Basically, when two FM/AMs cover some same addresses, they intersect. The bit-wise formula $(A_{\text{Mask}} + B_{\text{Mask}}) + \overline{A_{\text{Address}} \oplus B_{\text{Address}}}$ tells whether two FM/AMs intersect; only if the formula outputs all "1"s, A and B intersect. The intersection I between A and B is obtained using (1). Fig. 6(a) shows an example of calculating INTERSECT(A, B):

$$I_{\text{Mask}} = A_{\text{Mask}} \& B_{\text{Mask}} \qquad (1)$$
$$I_{\text{Address}} = A_{\text{Address}} + B_{\text{Address}}.$$

REMOVE is used to clear faults (FMs) or to block access (AMs) to certain addresses. For instance, after a rank repairing (i.e., replacing a faulty rank with a spare one), the faults in the rank address are cleared from MEMRES's database. Another example is that when a page is retired, the page address

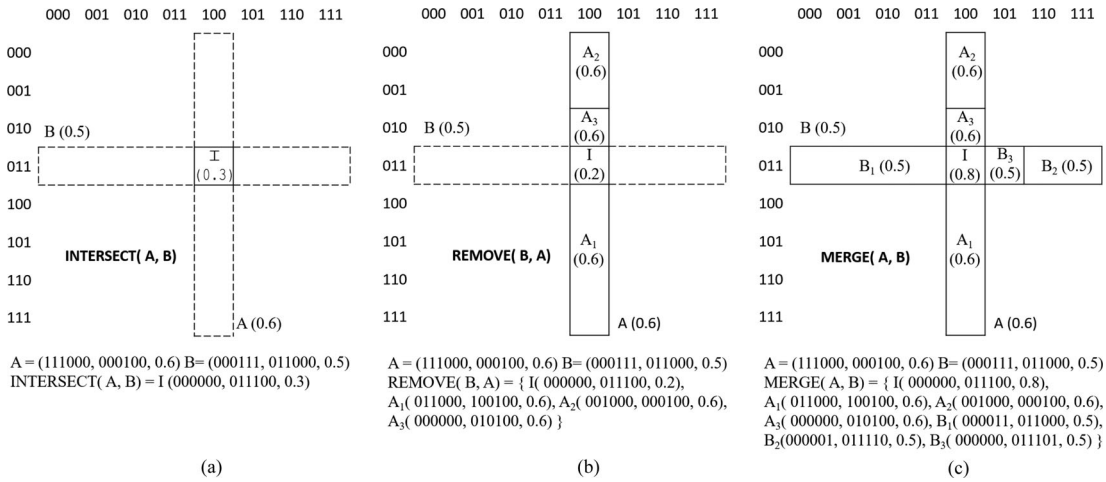


Fig. 6. Basic operations used in MEMRES: (a) INTERSECT, (b) MERGE, and (c) REMOVE. Cover-Rates of A and B are 0.6 and 0.5, respectively.

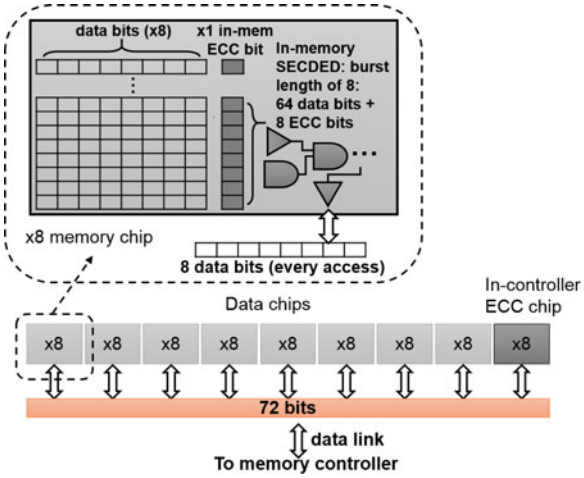


Fig. 7. Memory with in-memory SECDED and in-controller ECC. In a memory chip, every eight columns of data bits are protected by one column of ECC bits. The in-memory ECC logic can correct a single-bit error in a 72-bit ECC word (64 data and 8 ECC bits), where a burst length of 8 accesses is required for a $\times$ 8 chip to have 72 bits together for in-memory SECDED correction. A $\times$ 8 chip inputs/outputs 8 data bits in an access, and totally eight $\times$ 8 data chips and one $\times$ 8 ECC chip input/output 72 bits from/to the memory controller in an access, where data errors in the 72 bits can be corrected/detected by the in-controller ECC.

is removed from AMs and will never be accessed. Fig. 6(b) illustrates removing B from A. First, the intersection I between A and B is calculated, where the removing operation is actually only performed on I. Second, the Cover-Rate and Access-Rate of I are updated using (2), which represents the remaining intersection on A after removing B from it (if Access-Rate and Cover-Rate are zero or negative, I is totally removed from A). Third, Procedure 1 is used to obtain the least FM/AMs to cover the remaining parts of A excluding I. At last, store the remaining parts of A and the updat I together in a collection, which is the output of REMOVE(B,A):

$$I_{\mathrm{Access-Rate}} = A_{\mathrm{Access-Rate}} - B_{\mathrm{Access-Rate}} \tag{2}$$
$$I_{\mathrm{Cover-Rate}} = 1 - (1 - A_{\mathrm{Cover-Rate}}) / (1 - B_{\mathrm{Cover-Rate}}).$$

MERGE is used to combine two FM/AMs. For example, MEMRES merges and stores all accessed faulty addresses in its database so that if some addresses produce multiple errors, these addresses' Access-rates are more than 1 after being merged in database, and permanent faults are determined according to it, then MEMRES can perform reliability management to clean the detected permanent faults. In Fig. 6(c), a MERGE operation of A and B is illustrated. First, the intersection I between A and B is calculated, and then the Cover-Rate and Access-Rate of I are calculated following (3). Second, constructing the remaining parts of A and B after excluding I using Procedure 1. Finally, store I and remaining parts of A and B together in a collection:

$$I_{\mathrm{Access-Rate}} = A_{\mathrm{Access-Rate}} + B_{\mathrm{Access-Rate}} \tag{3}$$
$$I_{\mathrm{Cover-Rate}} = 1 - (1 - A_{\mathrm{Cover-Rate}}) \cdot (1 - B_{\mathrm{Cover-Rate}}).$$

E. Modeling

The increasing large amount of memory faults hurt the memory reliability significantly [1]–[4]. However, compared with intensive data access and massive memory operations, the memory faults and failures are still counted as rare events. Therefore, using traditional simulators to analyze memory reliability involves too many redundant computations like emulating memory operation and simulating fault propagation. In order to improve computation efficiency, MEMRES uses statistical models. In this section, we describe the statistical models for fault injection, ECC detection and correction, and implementation of memory reliability management.

1) Fault Injection: Memory faults are classified into two classes: transient and permanent faults. Intermediate faults can be modeled as permanent faults with a variable to model their existing time, which is not specially handled in MEMRES. Transient faults can be removed by written back after ECC correction, whereas permanent faults cannot be repaired. Field studies show that memory fault rate varies with the time [1], [2], [4] and locality [3]. However, traditional analytical models can only assume a constant fault rate, which results in inaccuracy. In MEMRES, the fault rate is held in AMs, which dynamically change with time and locality. In addition to the memory

Procedure 1: Remove FM/AM B from FM/AM A.

Input: FM/AM A and FM/AM B.
Output: The collection R of the remaining FM/AM in A after removing B

```
Set I = INTERSECT(A,B)
Set T = A
for i from the index of the MSB to the LSB of T_Mask do
    if T_Mask(i) == 1 && I_Mask(i) == 0 then
        //split T into two halves T0 and T1
        Set T_0, T_1 = T
        Set T_{0,Mask}(i) = 0 and T_{1,Mask}(i) = 0
        Set T_{0,Address}(i) = 0 and T_{1,Address}(i) = 1
        if T_0 intersects I then
            Add T_1 into the collection R. Set T = T_0
        else
            Add T_0 into the collection R. Set T = T_1
        end if
    end if
    if T == B then
        break loop
    end if
end for
return R
```

fault types studied in [1] and [2] (e.g., bit/row/column/bank faults, etc.), MEMRES also models single-bit transient errors (data-link errors and write/retention errors of emerging memories), which have high occurring rate and are easier to correct, but may dominate memory failure when they intersect with memory faults.

As an AM is valid through a simulation interval (i.e., AMs can be changed in different intervals), fault FIT rate (injection rate) is assumed to be constant in a simulation interval. Memory fault injection follows continuous Poisson distribution [30], and MEMRES describes the injection probability of k faults of a fault type in an interval in (4). It is noticed that the model is calculated once for every fault in an interval due to the different FIT rates of fault types. The injection model is more accurate than FaultSim, which assumes maximum one fault is injected in an interval:

$$P(k) = \frac{(\lambda \cdot t_{\mathrm{Int}})^k \cdot \exp(-\lambda \cdot t_{\mathrm{Int}})}{k!} \tag{4}$$

where λ is the failure rate (FIT rate) of a fault in an AM, and t_{Int} is the simulation interval in the unit of 10^9 h.

Data-link has a much higher bit error rate (BER) [8], [9] than memory faults due to intersymbol interference, signal reflection, clock jitter, and voltage variation. Data-link error is a transient error and only affects single bit in a 72-bit word (64 data bits and 8 ECC bits), which is easily corrected by ECC. Similarly to data-link error, write/retention errors of emerging memories, e.g., write failure in STT-RAM [11], [31], are also single-bit transient errors. As the probability of more than one single-bit transient errors simultaneously occurring in a word is extremely low, MEMRES safely assumes that there is at most one single-bit transient error in a ECC word (i.e., the ECC word length depends on algorithms and is commonly 72/144 bits). In a memory system with ECC and memory scrubbing (i.e., scrubbing periodically scans and corrects transient faults in memory), the single-bit transient errors cannot accumulate and may only cause memory system failure when its occurrence intersects with other memory faults in a word (i.e., occurs in the memory address as row/column/bank faults, etc.). Hence, to improve computation and memory consumption efficiency in a memory with ECC and scrubbing, MEMRES only injects single-bit transient errors with accessed memory faults. More specifically, the injection rate depends on the number of memory errors, which are produced when memory faults are accessed. As the data-link and write errors have even probability to occur at every access, these errors follow discrete Poisson distribution, where the probability of injecting k such errors intersected with a memory fault is described in (5). Differently, the retention errors have even temporal occurring probability, which follows continuous Poisson distribution and is handled in (4). For a memory without ECC or memory scrubbing, these single-bit transient errors are injected the same as single-bit transient faults, which are individually injected according to their FIT rates and can accumulate to create multiple-bit faults. However, such memory cannot functionally operate for a long period under high volume of single-bit transient faults:

$$P(k) = \frac{(\mathrm{BER} \cdot N_A)^k \cdot \exp(-\mathrm{BER} \cdot N_A)}{k!} \tag{5}$$

where BER is the BER of single-bit transient errors, and N_A is the expectation of number of accessed bits from memory faults in current simulation interval, which is calculated by Access-Rate and Cover-Rate of AMs and FMs.

2) ECC Algorithms and Designs: Common ECC algorithms are classified into two types: Hamming-based and symbol-based codes. Examples of Hamming-based codes are SECDED [32], [33] and SCCDCD (which can also be implemented by symbol-based codes) [34]. Both of them add 12.5% to redundancy of ECC bits. SECDED adds 8 bits (one $\times$ 8 chip or two $\times$ 4 chips) to a 64-bit word (eight $\times$ 8 chips or 16 $\times$ 4 chips), and SCCDCD adds 16 bits (four $\times$ 4 chips) to 128 bits (32 $\times$ 4 chips), which decodes/encodes two words interleaved across two channels simultaneously. Symbol-based codes are more sophisticated and efficient in redundancy like DDDC [12], [13]. With the same overhead of ECC bits as SCCDCD, DDDC has one sparing $\times$ 4 chip for failed chip replacement in addition to the function of SCCDCD. An extension of DDDC is that the second failed chip replacement is allowed, after which the ECC algorithm changes from SCCDCD to SECDED. These codes that can correct any errors from single chip are also called Chipkill. In MEMRES, an ECC algorithm is configurable with four parameters: maximum detectable faulty bits, maximum correctable faulty bits, maximum detectable faulty symbols (bits from a chip are a symbol, e.g., a 4-bit symbol for a $\times$ 4 chip), and maximum correctable faulty symbols. For SECDED, the maximum correctable and detectable faulty bits are one and two, respectively. For the extended DDDC, the initial maximum correctable and detectable faulty symbols are one and two respectively, but after replacing two failed chips, these two

parameters change to 0, whereas the maximum correctable and detectable faulty bits change to one and two, respectively. The overhead of density, delay, and power is outside the scope of this paper.

MEMRES allows two ECC designs: in-controller ECC and in-memory ECC, which are shown in Fig. 7. In-controller ECC designs are commonly used in commercial server-class CPUs, where ECC detection/correction logic circuits locate inside a memory controller and can correct and detect errors from both memories and data links (e.g., double data rate (DDR) buses). An in-memory ECC design locates in a memory chip and corrects errors inside the chip. As an example, the memory in Fig. 7 is constructed by $\times$ 8 chips, each chip inputs/outputs 8 bits in an access, and totally 72 bits from eight data chips and one ECC chip are read/written simultaneously, which comprise a word and are decoded/encoded by an in-controller ECC. In-memory ECC designs require burst mode. In burst mode, memory reads/writes multiple words with continuous physical addresses in one time. In Fig. 7, the in-memory SECDED works with burst mode with the burst length of 8. In every chip access, the ECC decodes/encodes 72 bits from eight continuous chip-words together to perform SECDED. As an in-controller SCCDCD fails when it faces multiple faulty symbols (chips), an in-memory ECC significantly decreases the probability of such case by correcting symbols inside chips.

The model for memory with only in-controller ECC design was derived in [7]. However, memories with both in-memory and in-controller ECC designs have not been studied. We derived a set of statistical models used in MEMRES's ECC check, which allow for the interaction of in-memory and in-controller ECCs and give the probability that a memory fault or intersection (represented by an FM) fails both in-memory and in-controller ECCs in a simulation interval. Currently, in-memory ECC only considers SECDED, and in-controller ECC allows all Hamming-based and symbol-based algorithms.

As an example, we show a model for the combination of an in-memory SECDED and an in-controller symbol-based code (e.g., SCCDCD). This model describes the probability that after a memory fault FM is injected, the FM fails both in-memory and in-controller ECC in a simulation interval.

Equation (6) calculates the probability ($P_{\text{correct\_symbol}}$) that a symbol (e.g., 4 bits for a $\times$ 4 chip) covered by an FM is ECC correctable. It includes two cases: 1) there is no faulty bit in the symbol ($P_{0\_\text{faultybit@symbol}}$) and 2) there is one faulty bit in the symbol, and it is the only one in its burst group of 72 bits (BL $\cdot$ N_{PFB}), which is correctable to SECDED ($P_{1\_\text{faultybit@symbol}}$).

$$\begin{aligned} P_{0\_\text{faultybit@symbol}} &= (1-P_{\text{FB}})^{N_{\text{PFB}}} \\ P_{1\_\text{faultybit@symbol}} &= C_1^{N_{\text{PFB}}} \cdot P_{\text{FB}} \cdot (1-P_{\text{FB}})^{BL\cdot N_{\text{PFB}}-1} \\ P_{\text{correct\_symbol}} &= P_{0\_\text{faultybit@symbol}} + P_{1\_\text{faultybit@symbol}}. \end{aligned} \tag{6}$$

Here, BL is the burst length (e.g., burst length of 8 is shown in Fig. 7). N_{PFB} is the number of possible faulty bits in the symbol, which is determined by the Mask of the FM covering the symbol. $C_1^{N_{PFB}}$ is choosing one faulty bit from N_{PFB} possible faulty bits in the symbol. P_{FB} is the probability that a bit covered by the FM is faulty, which is calculated from Cover-Rate of the FM.

Then, we calculate the probability ($P_{\text{correct\_word}}$) that a word constructed by possible faulty symbols is ECC correctable in (7) based on $P_{\text{correct\_symbol}}$ and number of possible faulty symbols N_{PFS} (calculated from Mask) in a word. The appearance of uncorrectable word will cause both memory failure

$$\begin{aligned} P_{\text{correct\_word}} = &\sum_{k=0}^{\min(N_{\text{MFS}},N_{\text{PFS}})} C_k^{N_{\text{PFS}}} \\ &\cdot {P_{\text{correct\_symbol}}}^{N_{\text{PFS}}-k} \cdot (1-P_{\text{correct\_symbol}})^k \end{aligned} \tag{7}$$

where the N_{MFS} is the maximum correctable symbols of the in-controller ECC, and k is the number of faulty symbols.

The model above excludes the data-link error and write/retention error, which may intersect with memory errors to create uncorrectable errors. The additional occurrence of a write/retention error inside a memory chip may cause failure in the in-memory SECDED which is otherwise able to correct the memory chip, and then the appearance of an additional faulty symbol may further cause failure in the in-controller ECC. As explained in Section II-E1, there is maximum one such error in a symbol, and hence we derive the probability ($P_{\text{wrErr\_fail\_memECC}}$) that the occurrence of a write/retention error in an accessed symbol causes an in-memory SECDED failure in (8). The derivation includes two cases: 1) a memory faulty bit already exists in the accessed symbol before the occurrence of a write/retention error in the symbol, and the write/retention error should not overlap with the faulty bit. 2) A memory faulty bit is not in the accessed symbol but in the same burst group (e.g., the 63 bits excluding the accessed symbol in a burst group, see Fig. 7) before the occurrence of a write/retention error in the accessed symbol:

$$\begin{aligned} P_{\text{wrErr\_fail\_memECC}} = &\left(\text{SL}-1\right)/\text{SL} \\ &\cdot P_{1\_\text{faultybit@symbol}} + (\text{BL}-1)\cdot N_{\text{PFB}} \cdot P_{\text{FB}} \\ &\cdot (1-P_{\text{FB}})^{(BL-1)\cdot N_{\text{PFB}}-1} \cdot P_{0\_\text{faultybit@symbol}} \end{aligned} \tag{8}$$

where SL is the symbol length (number of bits per symbol).

Then, the in-controller ECC may fail due to the additional faulty symbol caused by the write/retention error in the case that existing faulty symbols already reach the maximum correction capability N_{MFS} before the write/retention error. This probability $P_{\text{wrErr\_fail\_ctrECC}}$ is derived as follows:

$$\begin{aligned} P_{\text{wrErr\_fail\_ctrECC}} = &C_{N_{\text{MFS}}}^{N_{\text{PFS}}} \cdot (1-P_{\text{correct\_symbol}})^{N_{\text{MFS}}} \\ &\cdot C_1^{N_{\text{PFS}}-N_{\text{MFS}}} \cdot P_{\text{wrErr\_fail\_memECC}} \\ &\cdot {P_{\text{correct\_symbol}}}^{N_{\text{PFS}}-N_{\text{MFS}}-1}. \end{aligned} \tag{9}$$

Unlike write/retention errors, which occur and can be corrected inside memory chips, data-link errors occur on data links and are not checked by in-memory ECC. More specifically, when the existing faulty symbols reach the correction capability N_{MFS}, a data-link error occurring on any correct symbols will create another faulty symbol to fail in-controller ECC. The

probability is derived in the following equation:

$$P_{\mathrm{dlErr\_fail\_ctrECC}} = \left(N_S - N_{\mathrm{MFS}}\right)/N_S \cdot C_{N_{\mathrm{MFS}}}^{N_{\mathrm{PFS}}} \cdot P_{\mathrm{correct\_symbol}}{}^{N_{\mathrm{PFS}}-N_{\mathrm{MFS}}} \cdot \left(1 - P_{\mathrm{correct\_symbol}}\right)^{N_{MFS}}. \quad (10)$$

Here, N_S is the number of symbols in an ECC word (e.g., 32 for SCCDCD with $\times$ 4 chips).

So far, we have derived the statistical models for probabilities of uncorrectable words caused by a memory-fault, by the interaction of memory-fault and write/retention-error, and by the interaction of memory-fault and data-link-errors. These models also work for error detection. For memory faults with faulty bits exceeding the specified ECC detectable capability, ECC can still detect a part of them. For example, some errors with three faulty bits can be detected by SECDED that guarantees to detect faults with two or less faulty bits. A partial detection rate is taken as a constant input to model the detection probability of those faults.

3) Memory Reliability Management: Although ECC designs can correct memory errors, permanent faults can accumulate with time and intersect to produce uncorrectable multiple-bit and multiple-symbol errors. To avoid fault accumulation, modern systems use memory reliability management to deactivate existing faults. These techniques require information of fault location, which can be identified from detected errors. MEMRES has modeled the identification process. One collection of FMs called fault-collection stores all faulty addresses, which MERGE all existing faults. Errors are produced when fault-collection is accessed by AMs (i.e., AMs intersect with FMs in fault-collection), and then detected by ECC designs. MEMRES MERGE detected errors (also represented by FMs) into a collection called error-log. A high Access-Rate of an FM in error-log means that the addresses covered by the FM frequently produce errors and are identified as a permanent fault. This can trigger memory reliability management to deactivate it. Modeling of memory reliability management is detailed below:

1) Memory scrubbing corrects detected correctable transient faults. Two scrubbing techniques are simulated in MEMRES,
   a) if an ECC correctable transient fault (in the form of FM) is accessed by an AM, the accessed faulty addresses are removed from the transient fault using REMOVE;
   b) in addition to the on-line correction, the memory scrubbing periodical inspects the memory, and all correctable transient faults are cleared from fault-collection using REMOVE, where the periodically scrubbing cycle is configurable.
2) Hardware sparing allows the replacement of failed hardware with sparing hardware. For example, modern memory systems have enabled rank sparing, which uses a spare rank for replacing an in-use rank with detected permanent faults. In MEMRES, the spare devices, number of spare devices, the hardware being protected by this technique, and triggering threshold can be specified in configuration file. When detected permanent faults reach the configurable threshold in a protected hardware, REMOVE is applied to clear faults (in the form of FMs) in the replaced hardware from MEMRES's database (fault-collection and error-log).
3) Memory page retirement blocks access to memory pages with permanent faults to avoid fault activation. More specifically, when a permanent fault is detected in a memory physical page, the address mapped to the page is blocked, and the data on the page is moved to other pages. In MEMRES, once a permanent fault (in the form of FM) is detected by error-log, the access (in the form of AMs) to the pages intersecting with the fault are moved to other pages using REMOVE and MERGE. The maximum number of retired pages is configurable.
4) Memory mirroring mounts one memory space as a copy of another one, and system reads/writes data from/to both spaces simultaneously so that if data from one space contains errors, system can still obtain correct data from the other memory space. This protection is frequently used for critical data. MEMRES models this technique by having a special ECC check: uncorrectable faults in the mirrored spaces cannot directly cause failure unless two uncorrectable faults from two mirrored space intersect.

F. Tradeoff Between Accuracy and Speed

In this section, we analyze the accuracy-speed tradeoff in fault simulation. In MEMRES, dense grids of memory space (smaller AMs) more accurately models access behavior by holding more precise Cover-Rates, whereas sparse grids (larger AMs) save computation but lead to imprecise Cover-Rates. For example, one bank is intensively accessed but other banks in the same rank are barely accessed, if a large AM is used to represent access to the rank, its Cover-Rate is low after averaging memory access over all banks. Then, inaccurate simulation is resulted when a fault occurs in the intensively accessed bank, because the fault is supposed to be quickly activated (accessed) but the low Cover-Rate delays its activation time. Fault activation is a random event and determined by Cover-Rate, hence distribution of the activation time, which is in fact simulated in MEMRES, is crucial to accuracy. The mathematical expectation and variance of fault activation time are listed below:

$$\mu_{\mathrm{ActTime}} = t_{\mathrm{Int}} \cdot \frac{\left(1 - \textit{Cover-Rate}\right)^{\mathrm{MapSize}}}{1 - \left(1 - \textit{Cover-Rate}\right)^{\mathrm{MapSize}}} \quad (11)$$

$$\sigma_{\mathrm{ActTime}} = t_{\mathrm{Int}} \cdot \frac{\left(1 - \textit{Cover-Rate}\right)^{\mathrm{MapSize}}}{\left(1 - \left(1 - \textit{Cover-Rate}\right)^{\mathrm{MapSize}}\right)^2}.$$

Here, t_{Int} is the simulation interval, *Cover-Rate* is the Cover-Rate product of a fault FM and the AM accessing the FM, and $MapSize$ is the number of addresses in the intersection of the FM and AM. The accuracy is determined by the precision of *Cover-Rate* and $MapSize$. For a high *Cover-Rate*, the sensitivity of accuracy to $MapSize$ is smaller, and vice versa. Therefore, if fault injection is dominated by large faults or memory is intensively and uniformly accessed, larger AMs can be used to speed up the simulation without losing accuracy.

Similarly, too prolong simulation intervals can lead to less computation. Nevertheless, longer interval may result in simulation error if too many faults are injected in a long interval. For example, in a system with memory reliability management, a detected permanent fault should be deactivated by repairing techniques before intersecting with other faults occurring later. However, a long simulation interval increases the probability of multiple fault injection, which can intersect to produce non-correctable errors. Therefore, it had better avoid multiple fault injection, and the interval of hours is acceptable according to fault rates reported in [1] and [2].

The run time and memory consumption complexity of MEMRES are $O\left(\lambda^2 \cdot \log\left(M\right)^2\right)$ and $O\left(\lambda \cdot \log\left(M\right)\right)$, respectively, where M is the size of memory system and λ is the failure rate. The size and number of FM/AMs scale with the address length ($\log(M)$) and injection rate (λ), respectively. Run time of INTERSECT scales with the size of FM/AMs ($\log(M)$). The run time of MERGE and REMOVE (described in Procedure 1) is determined by the size of FM/AMs ($\log(M)$) and the run time of INTERSECT ($\log(M)$), which is $\log\left(M\right)^2$. The number of operations is proportion to the square of the number of FM/AMs (proportional to λ^2). The complexity of total run time, which is the product of the number of operations and the run time of operations, is $O\left(\lambda^2 \cdot \log\left(M\right)^2\right)$. The memory consumption, which is decided by the number and size of FM/AMs, is $O\left(\lambda \cdot \log\left(M\right)\right)$. In experiments, the wall time for the 100 000 5-year simulations (in-controller ECC and memory scrubbing are enabled) for Figs. 10 and 11 on Quad-Core AMD Opteron Processor 2376 with eight threads is 70 min. Its peak memory consumption is 1 GB.

III. Framework Validation

To substantially validate MEMRES is nontrivial. Large-scale experimentation is too expensive and impractical for most research groups given the need to observe failures over an extended period of time on a large-scale datacenter. Existing simulators take unacceptable time to complete the validation task, and existing analytical models do not support memory reliability management. In this section, we validate MEMRES's accuracy with FaultSim, the analytical model [7], and derived analytical models. Because models and FaultSim do not support fault rate distribution and variation over time and address space, memory access behavior, in-memory ECC, and memory reliability management, we disable these features in MEMRES in this section. However, enabling these features strongly affects simulation results, which are illustrated in Section IV. In the validation, we assume that fault rate is constant, a large fault affects all covered address space, and a fault is accessed and corrected or causes ECC failure immediately after the fault injection.

We use a 4-GB DRAM based main memory as an example to validate MEMRES. The configuration of the 4-GB DRAM is listed in Table II. In this memory architecture, an example column FM's mask is shown in Fig. 8. We use the fault rates reported in [1], [2], [8] and [9] for the validation, which is shown in Table III. Because analytical models [7] and FaultSim [17] assume all bits in a fault coverage are faulty, we use Cover-Rates of 1 for MEMRES to match their assumption in the validation for all faults (i.e., the Cover-Rate listed in the table are used for later case study in Section IV).

TABLE II
Architecture of a 4-GB DRAM DIMM

Ranks	Chips	Banks	Mats	Rows	Columns	Access-Rate
2	16 + 2	8	64	512	4096	1e12/h

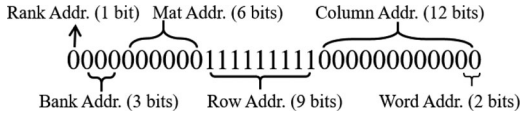


Fig. 8. Example Mask of a column FM in the memory specified by Table II. A read contains 4 bits from a chip, 18 reads construct a 72-bit word (64 data bits and 8 ECC bits), and a word has a memory physical address.

TABLE III
Fault FIT Rates Per Chip and Data-Link BER for DRAM and STT-RAM

Fault types	Transient FIT	Permanent FIT	Cover-Rate
Single-bit	0	18.6	1
Single-word	1.4	0.3	1
Single-column	1.4	5.6	0.02
Single-row	0.2	8.2	0.002
Single-bank	0.8	10	0.002
Multi-banks	0.3	1.4	0.002
Single-lane	0.9	2.8	0.002
Data-link BER	10^{-14} [9]		
Retention BER/h (STT-RAM)	$0, 10^{-5}, 10^{-10}, 10^{-15}$, (design dependence)		
Write BER (STT-RAM)	$0, 10^{-8}, 10^{-11}, 10^{-14}$ (design dependence)		

The retention BER (RER) and write BER (WER) are only for STT-RAM. Data-link error, retention error, and write error are single-bit transient errors (SBT).

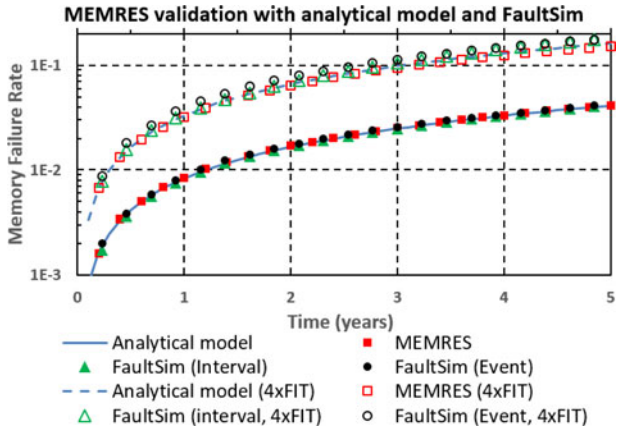


Fig. 9. Validation of MEMRES with FaultSim and the analytical model [7]. The failure rates for a 4-GB DRAM with SECDED as functions of time are shown. 1× and 4× fault rates are used. MEMRES matches well with the analytical model and FaultSim.

The predicted failure rates of a 4-GB DRAM by MEMRES, FaultSim (interval and event modes), and the analytical model [7] are drawn in Fig. 9 as a function of time. Normal fault

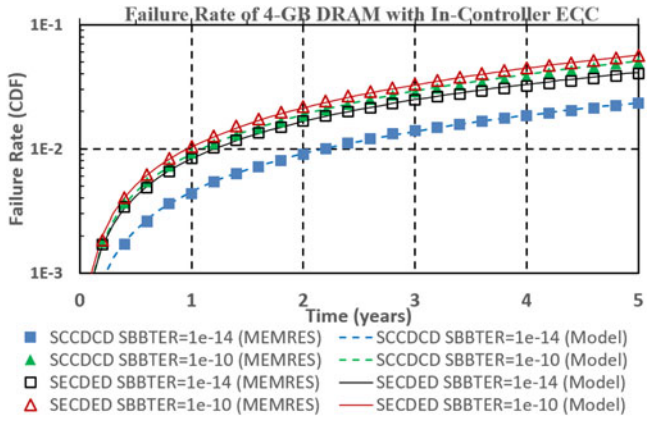


Fig. 10. Failure rate for a 4-GB DRAM with SECDED or SCCDCD as a function of time. The SBTERs (i.e., DRAM only has data-link error as SBT in the validation) of 10^{-14} and 10^{-10} are used in this validation.

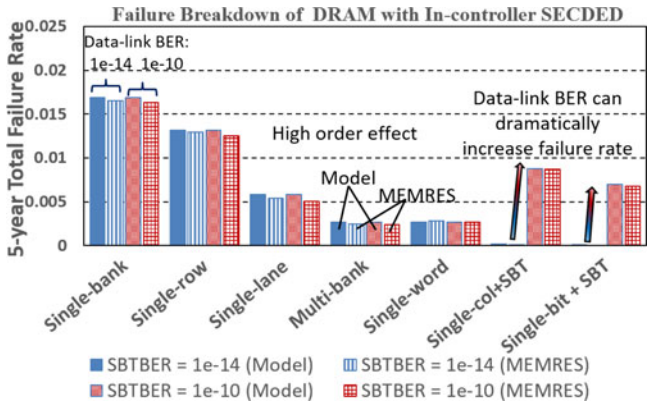


Fig. 11. Memory failure rate breakdown for a 4-GB DRAM operating for 5 years with in-controller SECDED. The data-link BER of 10^{-14} and 10^{-10} are used in this validation.

rate (Table III) and 4× fault rate (4× FIT) are used. Using the analytical model [7] as the baseline, the maximum mismatch for MEMRES, FaultSim's interval and event modes are 1%, 2%, and 2%, respectively. Please note that the analytical model is not the ground truth. The analytical model should overestimate the memory failure rate, because it separately calculates the probability of all critical faults that can cause memory failure and sums them together, but ignores to subtract the case that more than one critical faults exist in the same memory, which is the second order probability. Since MEMRES predicts lower failure probability than the analytical model indicating that MEMRES' error is lower than 1%.

In the following, we validate a DRAM with high frequency transient faults, e.g., DDR bus errors, which are not supported in FaultSim, and the analytical model [7]. These errors strongly affect memory reliability through intersecting with permanent memory faults to cause ECC failure, e.g., ChipKill. To fill the gap, we derive an analytical model for memory failure caused by the intersection between a memory permanent fault and a transient fault (applicable to all transient faults and single-bit transient errors) as a supplemental model to validate MEMRES. Intersection between two transient fault/errors is unlikely given that a transient fault only exists short time in memory with scrubbing, hence, we ignore this case.

P_{fail} in (12) is the probability that memory failure is caused by an intersection of one memory fault and one transient fault as a function of time. The model describes that firstly a permanent fault occurs ($P_{\text{perm}}(t_1)$) at time t_1, then a transient fault occurs ($P_{\text{perm}}(t_2)$) within $t - t_1$ after t_1, and they intersect with each other to produce uncorrectable errors ($P_{\text{intersect}}$), e.g., the two faults occur in different chips but cover overlap addresses to produce multiple-symbol errors to cause Chipkill failure. $P_{\text{intersect}}$ depends on fault size and ECC algorithms:

$$\begin{aligned} P_{\text{fail}}(t) &= P_{\text{intersect}} \cdot \int_0^t P_{\text{perm}}(t_1) \cdot \int_0^{t-t_1} P_{\text{tran}}(t_2) \quad (12) \\ P_{\text{perm}}(t_1) &= \exp(-\lambda_p t_1) \cdot (1 - \exp(-\lambda_p dt_1)) \\ &= \lambda_p \cdot \exp(-\lambda_p t_1)\, dt_1 \\ P_{\text{tran}}(t_2) &= \lambda_t \cdot \exp(-\lambda_t t_2)\, dt_2 \end{aligned}$$

where λ_p and λ_t are the failure rates of the permanent and transient fault, respectively. The analytical form of P_{fail} is shown as

$$P_{\text{fail}}(t) = P_{\text{intersect}} \cdot \left(\begin{array}{c} 1 - \exp(-\lambda_p t) + \lambda_p / (\lambda_t - \lambda_p) \\ \cdot (\exp(-\lambda_t t) - \exp(-\lambda_p t)) \end{array} \right). \quad (13)$$

For the cumulative distribution function (CDF) of memory failure rate shown in Fig. 10, MEMRES matches with the analytical model for a 4-GB DRAM with in-controller SECDED and SCCDCD. SCCDCD overall performs better than SECDED because the SCCDCD can correct multiple-bit errors from any single chip, indicating that SCCDCD can correct all individual faults from Table III. However, intersections of multiple faults from different chips and intersection between memory faults and single-bit transient errors (SBT, including data-link error, retention error of STT-RAM, and write error of STT-RAM) can give rise to SCCDCD failure. The failure rate increases dramatically with data-link BER, and when the BER is 10^{-10}, SCCDCD does not show obvious benefit compared with SECDED .

Fig. 11 shows the breakdown of failure caused by different fault types. Again, MEMRES matches well with analytical models except for a small difference. MEMRES more accurately shows that all single-fault induced failure rates decrease with increased data-link BER (e.g., single-bank, single-row, single-lane, multi-bank, and single-word), whereas the analytical model gives exact the same failure rates at different BER. In reality, as more memory failures are caused by the increased data-link errors interacting with permanent memory faults, failures caused by other memory faults decrease due to the fact that when memory system failure occurs, the system is shut down, and the failed memory is replaced with a new memory without any memory faults. Analytical models calculate the failure probability due to each fault individually because it is very difficult to include the high order effect of failure interactions. Note that, in default MEMRES setup (used in the paper), a memory

TABLE IV
THE 5-YEAR FAILURE RATE FOR STT-RAMS WITH DIFFERENT WRITE ERROR RATE (WER, PER-BIT-WRITE FAILURE PROBABILITY), RETENTION ERROR RATE (RER, PER-BIT-HOUR FAILURE PROBABILITY), AND ECC DESIGNS: 1) IN-CONTROLLER SCCDCD WITH (W/) IN-MEMORY SECDED, 2) IN-CONTROLLER SCCDCD WITHOUT (W/O) IN-MEMORY SECDED

RER	10^{-5}		10^{-10}		10^{-15}	
WER	W/O	W/	W/O	W/	W/O	W/
10^{-8}	0.1365	0.0219	0.1359	0.0219	0.1354	0.0209
10^{-11}	0.1336	0.0213	0.0649	0.0213	0.0638	0.0208
10^{-14}	0.1317	0.0213	0.0402	0.0212	0.0310	0.0207

failure ends simulations; however, MEMRES also models the failed memory replacement similarly to rank replacement (see Section II-E3), where MEMRES allows to continues simulation when a failure occurs and is fixed by hardware replacement.

IV. STUDY OF STT-RAM USING MEMRES

In this section, we perform a case study of analyzing STT-RAM's reliability using MEMRES. Since STT-RAM and traditional memories have similar peripheral circuits (sense amplifier, decoder, etc.), and large memory faults like row, column, bank faults, etc. are highly possible caused by peripheral circuit failure, we assume that STT-RAM suffers from the same fault FIT rates as DRAMs (see Table III) except for the single-bit transient fault (not single-bit transient errors) given that STT-RAM is immune to particle-induced faults. In addition to these faults, STT-RAM may suffer from single-bit transient errors including data-link errors, retention errors, and write errors. Their error rates depend on the STT-RAM design, where the error rates can be traded for low energy consumption [11].

There are tradeoffs between performance (write speed and energy) and reliability (retention error and write error) for STT-RAM. The critical current (i.e., proportional to write current) of STT-RAM is approximately proportional to thermal stability [11], whereas the thermal stability also determines the retention time, the average time that an STT-RAM cell holds data before false switching during standby state. This indicates that shorter retention time can reduce write energy as well as improve write speed at the risk of retention error [35]. Another tradeoff is to reduce write time at the expense of increased write error rate (WER) [36]. With memory reliability enhancement techniques, the reliability requirement of STT-RAM can be relaxed, which leads to faster speed and less power consumption simultaneously. As a case study, we use MEMRES to explore the impact of retention error rate (RER) and WER on STT-RAM with different reliability enhancement techniques. An 8-GB STT-RAM is used for all experiments in this section, which has two channels, and each channel has one dual DIMM with configuration in Table II. Fault FIT rates per chip are listed in Table III.

Table IV illustrates the 5-year memory failure rate of STT-RAMs with or without in-memory ECC for different write/retention BERs. The failure rate of the STT-RAM with both in-memory SECDED and in-controller SCCDCD is significantly lower than the STT-RAM with only in-controller

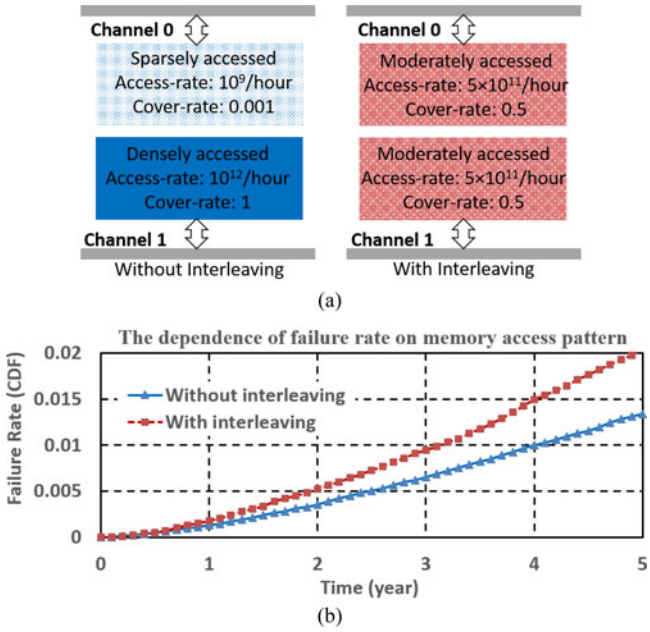


Fig. 12. (a) 8-GB STT-RAM with unbalanced memory access (without interleaving) and balanced memory access (with interleaving). (b) Failure rates (CDF) of STT-RAM with unbalanced and balanced memory access. In-controller SCCDCD and in-memory SECDED are enabled.

SCCDCD, because in-memory SECDED corrects a single-bit transient error soon after its occurrence, allows chips to output corrected symbols, and hence prevents multi-chip errors to cause in-controller SCCDCD failure. The probability of having multiple single-bit transient errors in a chip read (consequent 4 bits for a $\times$ 4 chips) is extremely low, which may not happen even once in a data-center for many years given that memory scrubbing is enabled to correct transient errors periodically. Therefore, for STT-RAM with in-memory ECC, retention and write errors can be traded for speed and power improvement, but seeking the optimized tradeoffs requires the knowledge of in-memory SECDED performance overhead.

As is known, memory interleaving can improve memory throughput by spreading memory access evenly across memory banks and channels. In Fig. 12(a), one STT-RAM with interleaving has balanced access, whereas the other one without interleaving has unbalanced access. In this study, all chips have the same memory fault FIT rates (see Table III). More specifically for this experiment, we assume fault occurrence probability (except single-bit transient errors) does not depend on memory access density for the reason that faults are random rare events and caused by process variation or particle induced errors which are not related to access density and hardware wear-out. The field study [4] shows a sublinear dependence of uncorrected error rate on time indicating that fault FIT rate does not increase with time and hence is not clearly related to memory access and wear-out (i.e., errors increase due to accumulated permanent faults, but fault occurring rate does not increase with time). Fig. 12(b) illustrates that the STT-RAM without interleaving suffers from lower failure rate given that permanent memory faults in the channel being sparsely accessed is less likely to intersect with data-link errors.

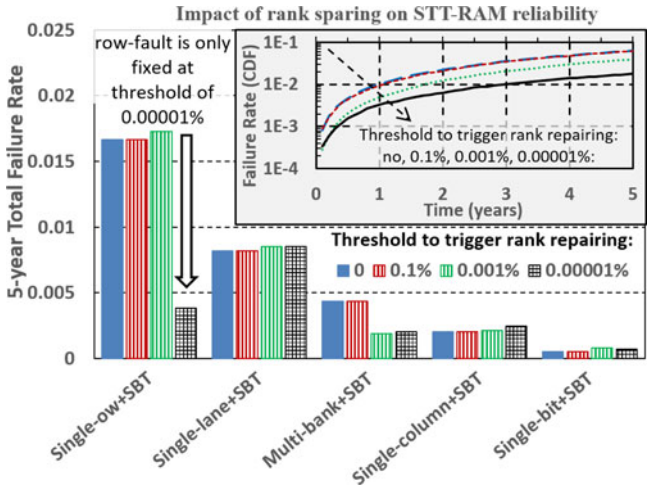


Fig. 13. Five-year failure breakdown and failure rate (CDF) of STT-RAM with enabled rank sparing. The thresholds (percentage of faulty addresses in a rank) to trigger rank repairing are 0.1%, 0.001%, and 0.00001%. The STT-RAM has one spare rank in each channel. In-controller SCCDCD is enabled.

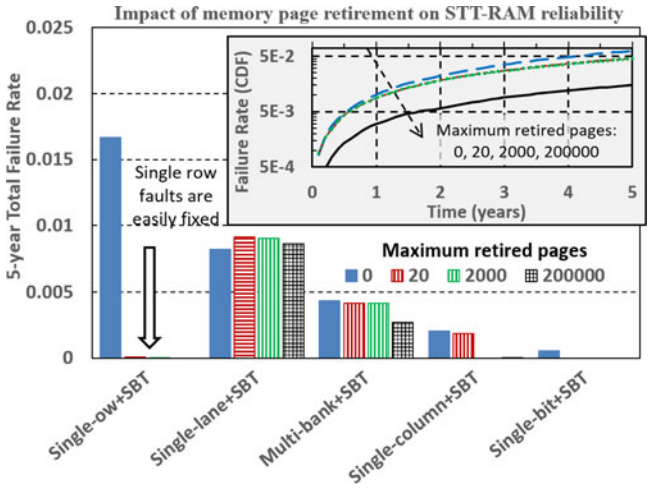


Fig. 14. Five-year failure breakdown and failure rate (CDF) of STT-RAM with enabled memory page retirement. Different maximum allowed retired pages per channel are tried, including 20, 2000, and 200 000. In-controller SCCDCD is enabled. Memory page size is 4 kB.

When rank sparing is enabled, detected permanent faulty addresses over a threshold can trigger rank sparing, which replaces the faulty rank with a spare rank. In Fig. 13, we simulates an 8-GB STT-RAM with rank sparing and different threshold to trigger rank sparing. As is illustrated, failure rate is not reduced when high threshold is set (e.g., 0.1%), because no individual fault is big enough to trigger rank sparing. With threshold decreasing, overall failure rate decreases given that multibank faults and single-row faults can trigger rank sparing at the threshold of 0.001% and 0.00001%, respectively. Single-lane fault is not correctable as it affects both ranks in a channel, whereas a channel only has one spare rank in the experiment setup.

A detected permanent fault can trigger memory page retirement, which removes faulty physical pages from use by the operating system. In Fig. 14, we simulate an 8-GB STT-RAM with memory page retirement. As can be seen, more allowed retired pages give rise to less failure rate but more memory space

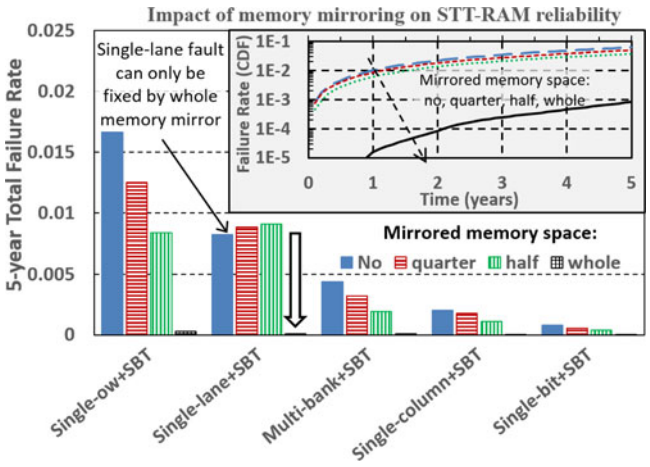


Fig. 15. Five-year failure breakdown and failure rate (CDF) of STT-RAM with memory mirroring. Different mirrored memory space are simulated including whole memory mirroring (one channel is mirrored to the other one), a half memory mirroring (one rank is mirrored to another one), and a quarter of memory mirroring (a half rank is mirrored to another half). In-controller SCCDCD is enabled.

loss. A row-fault is easy to be deactivated by retiring two pages, whereas a multibank fault affects more than 5000 pages and can only be deactivated when a large memory space loss is allowed. However, if rank sparing and memory page retirement are both enabled, they can collaborate to efficiently correct most faults.

A system with memory mirroring reads/writes two mirrored space simultaneously, and errors in one space can be corrected by its mirror. In Fig. 15, we simulate an 8-GB STT-RAM with memory mirroring, which mounts a memory space as a copy of another memory space. We tried different mirrored memory spaces including a whole memory space (i.e., a half-space is mirrored by the other half), a half-memory space, and a quarter of memory space. As mirrored space increases, the failure rates caused by all fault types decrease as is expected. Larger faults like lane faults require more mirrored space to correct. Memory mirroring significantly increases the system's robustness to memory faults, where failure rate decreases to nearly zero when a whole memory is mirrored, as a tradeoff, a half-memory space is lost.

Fault FIT rate varies over time and usually decreases with time [1], [2], whereas analytical models always assume constant FIT rate because a varying FIT rate is very difficult to model. However, constant FIT rate assumption may result in inaccuracy. As an example, we use MEMRES to simulate STT-RAMs with constant and varying FIT rats. In Fig. 16(a), the varying FIT rate is a quadratic function of time, which is normalized to the constant FIT rate such that it gives rise to the same 5-year-fault-occurring probability as the constant FIT rate. Though the varying FIT rate decreases with time, permanent faults continue to produce errors after injection; hence, error rates increase still with time, which is not contrary to field studies [1]–[4]. Fig. 16(b) shows that the varying FIT rate results in higher STT-RAM failure rate, because its higher FIT rate in the beginning causes faults to occur earlier, which have higher probability to intersect with data-link errors and fail SCCDCD ECC. The

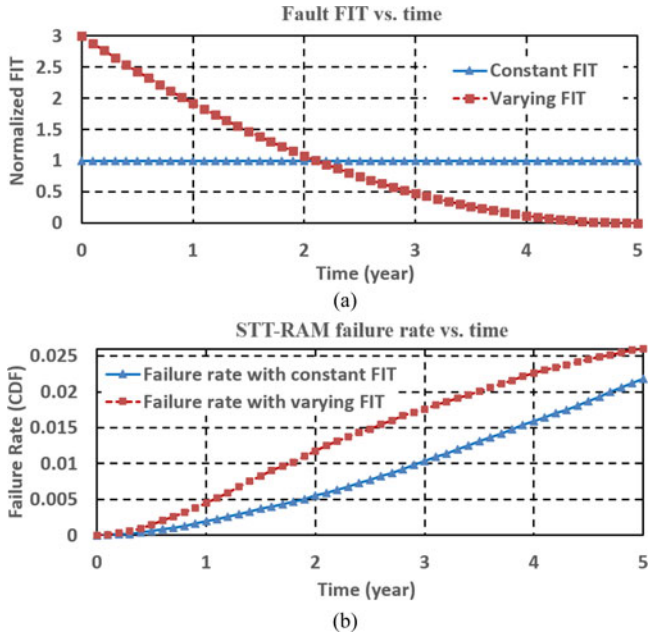


Fig. 16. (a) Varying fault FIT rate (normalized to constant fault FIT rate) and constant FIT rate versus time. (b) Failure rate (CDF) of STT-RAM with varying fault FIT rate and constant fault FIT rate (listed in Table III). In-controller SCCDCD and in-memory SECDED are enabled.

failure rate difference demonstrates MEMRES's capability of simulating more realistic situations than analytical models.

V. Conclusion

The proposed MEMRES framework facilitates a fast and convenient way to assess the reliability of modern memory systems. It can perform memory fault simulation with ECC and memory reliability management. The accuracy of MEMRES is validated by the comparison with the derived analytical model and existing models. Through MEMRES, modern reliability enhancement techniques including ECC designs and memory reliability management can be calibrated to have optimized efficiency for target applications. With additional fault models, we show examples of using MEMRES to optimize the reliability for emerging memory systems.

Acknowledgment

This work was conducted jointly between Samsung Semiconductor and the NanoCAD Lab of the Electrical Engineering Department, University of California, Los Angeles. The authors thank Dr. D. Niu of Samsung Semiconductor, M. Gottscho, and Dr. P. Nair for supporting this work while Mr. Wang was an Intern at Samsung Semiconductor in 2014. Funding came partly from the NSF Variability Expedition Grant No. CCF-1029783

References

[1] V. Sridharan and D. Liberty, "A study of DRAM failures in the field," in *Proc. 2012 Int. Conf. High Performance Comput. Netw. Storage Anal.*, Nov. 2012, pp. 1–11.

[2] V. Sridharan *et al.*, "Memory errors in modern systems: The good, the bad, and the ugly," in *Proc. 20th Int. Conf. Architectural Support Programming Languages Operating Syst.*, 2015, pp. 297–310.

[3] J. Meza, Q. Wu, S. Kumar, and O. Mutlu, "Revisiting memory errors in large-scale production data centers: Analysis and modeling of new trends from the field," in *Proc. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, 2015, pp. 415–426.

[4] B. Schroeder, E. Pinheiro, and W.-D. Weber, "DRAM errors in the wild: A large-scale field study," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 37, no. 1, pp. 193–204, 2009.

[5] W. Zhao *et al.*, "Failure and reliability analysis of STT-MRAM," *Microelectron. Rel.*, vol. 52, no. 9, pp. 1848–1852, 2012.

[6] J. Li, C. Augustine, S. Salahuddin, and K. Roy, "Modeling of failure probability and statistical design of spin-torque transfer magnetic random access memory (STT MRAM) array for yield enhancement," in *Proc. 45th ACM/IEEE Des. Autom. Conf.*, 2008, pp. 278–283.

[7] X. Jian, N. Debardeleben, S. Blanchard, V. Sridharan, and R. Kumar, "Analyzing reliability of memory sub-systems with double-chipkill detect/correct," in *Proc. 2013 IEEE 19th Pacific Rim Int. Symp. Dependable Comput.*, Dec. 2013, pp. 88–97.

[8] N. Miura, K. Kasuga, M. Saito, and T. Kuroda, "An 8Tb/s 1pJ/b 0.8 mm^2/Tb/s QDR inductive-coupling interface between 65 nm CMOS GPU and 0.1 μm DRAM," in *Proc. IEEE Int. Solid-State Circuits Conf.*, 2010, pp. 436–437.

[9] N. Nguyen *et al.*, "A 16-Gb/s differential I/O cell with 380 fs RJ in an emulated 40nm DRAM process," in *Proc. 2008 IEEE Symp. VLSI Circuits*, 2008, pp. 128–129.

[10] Y. Zhang, X. Wang, and Y. Chen, "STT-RAM cell design optimization for persistent and non-persistent error rate reduction: A statistical design view," in *Proc. Int. Conf. Comput.-Aided Des.*, 2011, pp. 471–477.

[11] S. Wang, H. Lee, F. Ebrahimi, P. K. Amiri, K. L. Wang, and P. Gupta, "Comparative evaluation of spin-transfer-torque and magnetoelectric random access memory," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 6, no. 2, pp. 134–145, Jun. 2016.

[12] X. Jian and R. Kumar, "Adaptive reliability chipkill correct (ARCC)," in *Proc. 2013 IEEE 19th Int. Symp. High Performance Comput. Arch.*, 2013, pp. 270–281.

[13] T. Willhalm, "Independent channel vs. lockstep mode—Drive your memory faster or safer," Santa Clara, CA, USA: Intel, 2014.

[14] C.-F. Wu, C.-T. Huang, and C.-W. Wu, "RAMSES: A fast memory fault simulator," in *Proc. Int. Symp. Defect Fault Tolerance VLSI Syst.*, 1999, pp. 165–173.

[15] T. M. Niermann, W.-T. Cheng, and J. H. Patel, "PROOFS: A fast, memory-efficient sequential circuit fault simulator," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 11, no. 2, pp. 198–207, Feb. 1992.

[16] W. Wu, F. Gong, G. Chen, and L. He, "A fast and provably bounded failure analysis of memory circuits in high dimensions," in *Proc. 2014 19th Asia South Pacific Des. Autom. Conf.*, 2014, pp. 424–429.

[17] P. J. Nair, D. A. Roberts, and M. K. Qureshi, "FaultSim: A fast, configurable memory-reliability simulator for conventional and 3D-stacked systems," *ACM Trans. Arch. Code Optim.*, vol. 12, no. 4, 2015, Art. no. 44.

[18] D. H. Yoon and M. Erez, "Virtualized and flexible ECC for main memory," *ACM SIGARCH Comput. Arch. News*, vol. 38, no. 1, pp. 397–408, 2010.

[19] M. Gottscho, C. Schoeny, L. Dolecek, and P. Gupta, "Software-defined error-correcting codes," *Proc. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, 2016, pp. 1–7.

[20] S. Tehrani, J. Slaughter, E. Chen, M. Durlam, J. Shi, and M. DeHerren, "Progress and outlook for MRAM technology," *IEEE Trans. Magn.*, vol. 35, no. 5, pp. 2814–2819, Sep. 1999.

[21] C. Heide, "Spin currents in magnetic films," *Phys. Rev. Lett.*, vol. 87, no. 19, 2001, Art. no. 197201.

[22] D. Worledge *et al.*, "Spin torque switching of perpendicular Ta-CoFeB-MgO-based magnetic tunnel junctions," *Appl. Phys. Lett.*, vol. 98, no. 2, 2011, Art. no. 022501.

[23] "ITRS," 2008, 2011. [Online]. Available: http://www.itrs.net/about.html

[24] E. Kultursay, M. Kandemir, A. Sivasubramaniam, and O. Mutlu, "Evaluating STT-RAM as an energy-efficient main memory alternative," in *Proc. 2013 IEEE Int. Symp. Perform. Anal. Syst. Softw.*, 2013, pp. 256–267.

[25] S. Wang, H. Lee, C. Grezes, P. Khalili, K. L. Wang, and P. Gupta, "MTJ variation monitor-assisted adaptive MRAM write," in *Proc. 53rd Annu. Des. Autom. Conf*, 2016, p. 169.

[26] W. Wu, S. Bodapati, and L. He, "Hyperspherical clustering and sampling for rare event analysis with multiple failure region coverage," in *Proc. 2016 Int. Symp. Phys. Des.*, 2016, pp. 153–160.

[27] S. Wang, G. Leung, A. Pan, C. O. Chui, and P. Gupta, "Evaluation of digital circuit-level variability in inversion-mode and junctionless finfet technologies," *IEEE Trans. Electron Devices*, vol. 60, no. 7, pp. 2186–2193, Jul. 2013.
[28] W. Kang, L. Zhang, J.-O. Klein, Y. Zhang, D. Ravelosona, and W. Zhao, "Reconfigurable codesign of STT-MRAM under process variations in deeply scaled technology," *IEEE Trans. Electron Devices*, vol. 62, no. 6, pp. 1769–1777, Jun. 2015.
[29] S. Wang, A. Pan, C. O. Chui, and P. Gupta, "Proceed: A pareto optimization-based circuit-level evaluator for emerging devices," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 1, pp. 192–205, Jan. 2016.
[30] "Failure rate," 2008, 2011. [Online]. Available: https://en.wikipedia.org/wiki/Failure_rate
[31] W. Kang *et al.*, "Yield and reliability improvement techniques for emerging nonvolatile STT-MRAM," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 5, no. 1, pp. 28–39, Mar. 2015.
[32] M. Blaum, R. Goodman, and R. McEliece, "The reliability of single-error protected computer memories," *IEEE Trans. Comput.*, vol. 37, no. 1, pp. 114–119, Jan. 1988.
[33] W. Mikhail, R. Bartoldus, and R. Rutledge, "The reliability of memory with single-error correction," *IEEE Trans. Comput.*, vol. 31, no. 6, pp. 560–564, Jun. 1982.
[34] T. J. Dell, "A white paper on the benefits of chipkill-correct ECC for PC server main memory," *IBM Microelectronics Division*, pp. 1–23, 1997.
[35] Z. Sun *et al.*, "Multi retention level STT-RAM cache designs with a dynamic refresh scheme," in *Proc. 44th Annu. IEEE/ACM Int. Symp. Microarchitecture*, 2011, pp. 329–338.
[36] X. Wang, W. Zhu, M. Siegert, and D. Dimitrov, "Spin torque induced magnetization switching variations," *IEEE Trans. Magn.*, vol. 45, no. 4, pp. 2038–2041, Apr. 2009.

Shaodi Wang (S'12) received the B.S. degree from the Division of Microelectronic, Electronics Engineering and Computer Science Department, Peking University, China, and the M.S. degree in electrical engineering from the University of California (UCLA), Los Angeles, CA, USA. He is currently a fifth-year Ph.D. degree student in the NanoCAD Lab, Department of Electrical Engineering, UCLA, advised by Prof. Puneet Gupta.

His research interests include emerging memory and device technology circuit- and system-level design, evaluation and optimization, and modeling for manufacturing.

.

Henry (Chaohong) Hu received the B.S. degree in electrical engineering from Zhejiang University, Hangzhou, China, in 2000, and the Ph.D. degree in microelectronics and solid state electronics from Shanghai Jiaotong University, Shanghai, China, in 2004.

From 2003 to 2004, he was a Visiting Research Fellow in the Computer Engineering Lab, Delft University of Technology, Delft, The Netherlands. He is a Senior Staff Memory System Architect in the System Architecture Lab, Samsung Semiconductor Inc., San Jose, CA, USA. Before joining Samsung, he spent 10 years in memory technology on multiple R&D roles at Intel and Micron. His current research work is new memory system architecture, memory RAS, etc.

Hongzhong Zheng (M'05) received the B.S. and M.S. degrees in electrical engineering and computer science from the Huazhong University of Science and Technology, China, in 1998 and 2001, respectively, and the Ph.D. degree in electrical and computer engineering from the University of Illinois, Chicago, IL, USA, in 2009.

He is currently a Memory and Storage System Architect in Samsung System Architecture Labs, San Jose, CA, USA. His research interests include computer architecture, energy-efficient computing system designs, novel memory architecture, emerging memory technology, and performance modeling.

Dr. Zheng is a Member of the ACM.

Puneet Gupta (M'03) received the B.Tech. degree in electrical engineering from Indian Institute of Technology, Delhi, India, in 2000 and the Ph.D. degree from the University of California, San Diego, CA, USA, in 2007.

He is currently a Faculty Member of the Electrical Engineering Department, University of California, Los Angeles, CA, USA. He co-founded Blaze DFM Inc. (acquired by Tela Inc.) in 2004 and served as a Product Architect until 2007. He currently leads the IMPACT+ Center, which focuses on future semiconductor technologies. His research has focused on building high-value bridges across application-architecture-implementation-fabrication interfaces for lowered cost and power, increased yield, and improved predictability of integrated circuits and systems. He has authored more than 150 papers, 17 U.S. patents, a book, and a book chapter.

Dr. Gupta received the NSF CAREER Award, the ACM/SIGDA Outstanding New Faculty Award, the SRC Inventor Recognition Award, and the IBM Faculty Award. See http://nanocad.ee.ucla.edu and http://impact.ee.ucla.edu for more details.