Citadel: Efficiently Protecting Stacked Memory from TSV and Large Granularity Failures

PRASHANT J. NAIR, School of Electrical and Computer Engineering, Georgia Institute of Technology
DAVID A. ROBERTS, AMD Research, Advanced Micro Devices Inc.
MOINUDDIN K. QURESHI, School of Electrical and Computer Engineering, Georgia Institute of Technology

Stacked memory modules are likely to be tightly integrated with the processor. It is vital that these memory modules operate reliably, as memory failure can require the replacement of the entire socket. To make matters worse, stacked memory designs are susceptible to newer failure modes (e.g., due to faulty through-silicon vias, or TSVs) that can cause large portions of memory, such as a bank, to become faulty. To avoid data loss from large-granularity failures, the memory system may use symbol-based codes that stripe the data for a cache line across several banks (or channels). Unfortunately, such data-striping reduces memory-level parallelism, causing significant slowdown and higher power consumption.

This article proposes *Citadel*, a robust memory architecture that allows the memory system to retain each cache line within one bank. By retaining cache lines within banks, Citadel enables a high-performance and low-power memory system and also efficiently protects the stacked memory system from large-granularity failures. Citadel consists of three components; *TSV-Swap*, which can tolerate both faulty data-TSVs and faulty address-TSVs; Tri-Dimensional Parity (3DP), which can tolerate column failures, row failures, and bank failures; and *Dynamic Dual-Granularity Sparing (DDS)*, which can mitigate permanent faults by dynamically sparing faulty memory regions either at a row granularity or at a bank granularity. Our evaluations with real-world data for DRAM failures show that Citadel provides performance and power similar to maintaining the entire cache line in the same bank, and yet provides 700× higher reliability than ChipKill-like ECC codes.

Extension of Conference Paper: The paper is an extension of "Citadel: Efficiently Protecting Stacked Memory from Large Granularity Failures" [Nair et al. 2014]. This submission adds the following items that are not present in the original paper:

—Section 5.5 describes TSV SWAP for Alternate Memory Organizations (HMC-Like and Tezzaron-Like).
—Section 5.6 describes a bucket and balls analysis for SET-based structure design TSV SWAP.
—Section 6 evaluates single-bit ECC such as SECDED with TSV SWAP.
—Section 7.5.4 describes the additional traffic behavior for workloads using 3DP with caching.
—Section 9 proposes a SEC-based optimization to reduce correction latency in the common case of single-bit errors.

These add more than 30% newer material in terms of giving greater insight into different stacked memory organizations, optimizing design of TSV SWAP, single-bit ECC schemes with TSV SWAP, analysis of memory traffic intensity for 3DP, and also proposes a novel technique to optimize correction latency of Citadel for the common case of single bit errors. We have also expanded all graphs in the Results section to include per workload results.

This work was supported in part by Center for Future Architectures Research (C-FAR), one of the six SRC STARnet Centers, sponsored by MARCO and DARPA.

Authors' addresses: P. J. Nair and M. K. Qureshi, School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332; emails: pnair6@gatech.edu, moin@ece.gatech.edu; D. A. Roberts, 1 AMD Place, Sunnyvale, California 94088; email: David.Roberts@amd.com.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© 2016 ACM 1544-3566/2016/01-ART49 $15.00

DOI: http://dx.doi.org/10.1145/2840807

CCS Concepts: • **Computer systems organization → Reliability**; **Availability**; • **Hardware → 3D integrated circuits**; **Semiconductor memory**; **Transient errors and upsets**; *System-level fault tolerance;*

Additional Key Words and Phrases: Reliability, system memory

ACM Reference Format:
Prashant J. Nair, David A. Roberts, and Moinuddin K. Qureshi. 2016. Citadel: Efficiently protecting stacked memory from TSV and large granularity failures. ACM Trans. Archit. Code Optim. 12, 4, Article 49 (January 2016), 24 pages.
DOI: http://dx.doi.org/10.1145/2840807

1. INTRODUCTION

The emerging 3D stacked DRAM technology can help with the challenges of power consumption, bandwidth demands and reduced footprint. One of the key enablers of stacked memory is the *through-silicon via* (TSV) technology, which makes it possible to cost-effectively stack multiple memory dies on top of each other [Kang et al. 2009]. The shorter internal data paths afforded by TSVs reduce capacitance and active power. By exploiting wide buses [Consortium 2013] or high-frequency SerDes interfaces [Standard 2013] and higher levels of internal parallelism, both bandwidth and random-access latency are improved. It is anticipated that high-performance stacked memories often will be permanently attached to host processors via direct stacking, silicon interposers, or other hard-wired interconnects. In such a system, memories that develop permanent faults must continue to work in order to avoid replacement of multiple chips which tends to be expensive. These factors motivate the adoption of a *fail-in-place* philosophy for designing stacked memory systems [Dubash 2004].

Recent work on DRAM reliability [Sridharan and Liberty 2012] showed that large-granularity DRAM chip failures, such as bank failures, occur nearly as frequently as single-bit failures in commodity DIMMs. Stacked memory designs would not only be subject to these failures but also to newer fault models, such as arising from faulty TSVs. TSV faults can cause failures of several dies, often manifested as column failures or bank failures. Thus, stacked memory systems will be more vulnerable to large-granularity failures. Unfortunately, conventional error correction schemes such as ECC DIMMs [Silicon Power 2010] are targeted toward correcting random bit errors and are ineffective at tolerating large-granularity faults. Memory systems can tolerate large granularity failures using symbol-based coding schemes like ChipKill [Dell 1997]. However, this increases the number of activated chips and total power consumption.

To optimize performance and power for stacked memory, we want to retain the data for a cache line within a single bank. However, a bank failure would then cause loss of data for the whole cache line. One can adopt a philosophy similar to ChipKill for tolerating large-granularity failures for stacked DRAM. In such a design, the data for a cache line would be striped across several banks (or channels), and a symbol-based coding can be applied, in which the size of each symbol would be equal to the amount of data stored in each bank. Unfortunately, such a data mapping would require the memory system to activate several banks to service a single request. This causes performance degradation (10% to 25%) due to loss of bank(channel)-level parallelism, and power consumption (as high as 6× in our evaluations) due to activation of several banks to service one request.

As shown in Figure 1, ideally we want a system that has the performance and power efficiency of storing the entire cache line in one bank (NoStripe), and yet maintains robustness to large granularity faults (Stripe). To that end, this article proposes *Citadel*, a robust memory architecture that allows the memory system to retain each cache line within one bank (delivering high performance and low power) and yet efficiently protects the stacked memory from large-granularity failures.

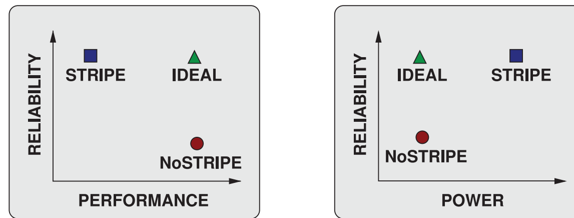

Fig. 1. Striping enhances reliability but sacrifices performance and power efficiency. Ideally, we want to tolerate large-granularity failures at high performance and low power.

Like ECC DIMMs, which have one additional chip per eight chips, in our study, Citadel has one extra die (ECC die) with smaller rows along with eight data dies. Similar to an ECC-DIMM that provides 64 bits of ECC for every 512-bit cache line, Citadel uses the 64 bits of metadata associated with each 512-bit cache line. Based on key insights, Citadel employs a three-pronged approach for fault tolerance.

Insight 1—Protect Against Runtime TSV Faults. As faulty TSVs tend to be a major cause of multi-bank failures in stacked memories, our first idea, *TSV-Swap*, specifically targets TSV faults that happen at runtime. DRAM vendors can use manufacture-level spare TSVs [Hsieh et al. 2010] to repair faulty TSVs at design time. Unfortunately, manufacture-level sparing does not protect against runtime failures. Citadel proposes TSV-SWAP, a technique that does not rely on any manufacturer-provided spare TSVs. Instead, TSV-Swap dynamically exchanges faulty TSVs with non-faulty TSVs with a remapping circuit. We found that while a data TSV typically affects only 1 bit in a data line (albeit across many lines), a failure of one of the address TSVs can make half of the memory unreachable. Thus, address TSVs are much more critical than data TSVs for system reliability. Our proposal, TSV-Swap, can repair up to eight faulty TSVs, which can be data, address, or command TSVs.

Insight 2—Detect and Correct Large-Granularity Failure. Even after mitigation of TSV related faults, the stacked memory is still vulnerable to internal DRAM die faults. We want to protect stacked memory not only from small-granularity failures (such as bit-fault or word-fault) but also from large granularity faults such as column-fault, row-faults, or even complete bank failures. Our second idea, *Tri-Dimensional Parity (3DP)*, provides highly effective and storage efficient correction for both small and large granularity failures. The 3DP proposal maintains parity in three dimensions: (1) across all banks and dies for individual rows, (2) across all rows in all banks within a die, and (3) Across all rows in single bank across all dies. Each line is equipped with CRC-32 [Peterson and Brown 1961] to detect data errors. If any error is detected, it is corrected using the parity information of 3DP. 3DP provides 130$\times$ higher resilience than just applying 2D-ECC. 3DP achieves this with only 1.6% storage overhead, compared to the 25% storage required for prior 2D schemes.

Insight 3—Isolate Faulty Memories with Efficient Sparing. When a fault is detected, data is restored using the correction capability of 3DP. However, modules with permanent faults would incur the correction overheads frequently. To avoid such frequent correction, we would like to redirect a faulty memory unit to a spare area. Unfortunately, if the sparing granularity is too fine, then it incurs significant tracking overheads (e.g., if a bank fails then thousands of rows get spared to the spare area). If the sparing granularity is too coarse, then it results in significant wasted space (e.g., sparing at a bank granularity would be wasteful if only one row is faulty). We make a key observation that a bank typically has either one or two row failures, or has thousands of row

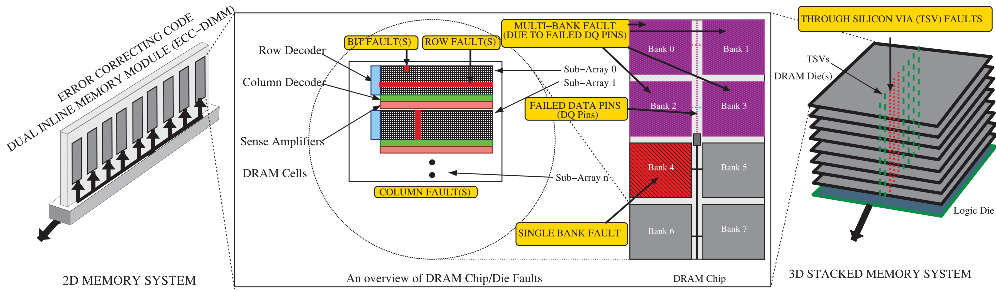

Fig. 2. Granularity of faults that occur in a DRAM Chip/Die. Faults can be at granularities of bit, column, row, bank(s), TSVs, and I/O links for stacked memory systems. Common wiring faults within a chip can cause multiple banks to fail.

failures (due to a sub-array or bank failure). Our third idea, *Dynamic Dual-Grained Sparing (DDS)*, exploits the bimodal behavior of faulty units and efficiently spares either at a row or bank granularity. Our proposed design of DDS can spare two faulty banks along with several row failures.

We perform reliability studies using real field data and perform sensitivity studies when field data is unavailable (e.g., for TSVs). Our evaluations, with an industry-grade fault simulator [David and Prashant 2014], shows that Citadel provides 100× to 1,000× higher reliability while still retaining power and performance similar to a system that maps the entire cache line in the same bank. Citadel achieves this using a storage overhead similar to ECC DIMMs (14% vs. 12.5%).

2. BACKGROUND AND MOTIVATION

Stacked memory systems have lower energy per bit and higher bandwidth when compared to their 2D counterparts. However, to obtain the power-efficiency and high bandwidth of stacked memory, the system must first address reliability challenges. As shown in Figure 2, failures can occur in a memory system at different granularities [Sridharan and Liberty 2012; Schroeder et al. 2009; Schroeder and Gibson 2010; Sridharan et al. 2013].

2.1. Memory Faults for Traditional Systems

A memory DIMM consists of multiple DRAM chips. A DRAM chip is organized into banks, where all banks share a common data bus. These banks are composed of rows and columns and are divided into sub-arrays. The banks contain row and column decoders that activate the word lines or select bit lines associated with the memory request. Faults at the DIMM level can affect all DRAM chips within a DIMM. However, the faults in individual chips are largely independent of each other. In this paper, the definitions for the chip faults follow that of Sridharan et. al. [Sridharan and Liberty 2012] and are represented in Figure 2. We note that banks are operated almost independently and share only wiring such as data, address, and command buses [Yoo et al. 1996; Shiratake et al. 1999]. Bank and rank faults occur mainly from faulty data or address or command buses.

2.2. Transposing Faults onto 3D Stacked Memories

Layout of an individual die in 3D stacked memory systems shows that its internal organization is very similar to that of a chip in conventional 2D memory systems [Kim et al. 2011; Pawlowski 2011; Hollis 2012; Bolaria 2011]. To a first order, this article transposes failure rates for all fault types except complete bank and complete rank for current 2D memory system onto stacked memory systems. The key difference is the

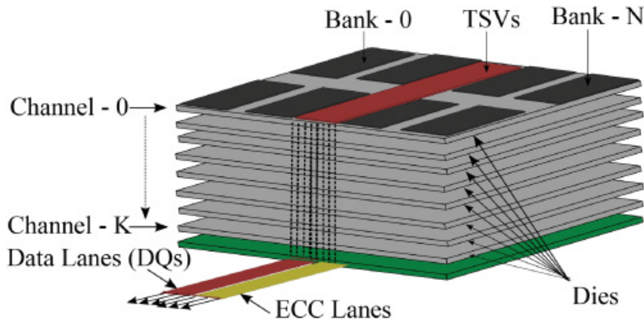

Fig. 3. HBM has a channel(s) per die and all banks in this channel are on the same die. HBM specification includes separate Data and ECC lanes.

introduction of TSVs for connecting data and address lines [Kang et al. 2009]. Due to this, complete bank faults and complete rank faults in any 3D stacked memory are now influenced by TSV faults.

2.3. Stacked Memory: Organization and ECC Layout

There are several design prototypes of stacked memory, including the High-Bandwidth Memory (HBM) [Standard 2013], Hybrid Memory Cube (HMC) [Hybrid Memory Cube Consortium 2013; Pawlowski 2011], and Octopus from Tezzaron [Tezzaron Semiconductor 2010]. These standards differ in their data organization and also share TSVs differently. However, these stacked memory systems fundamentally have the same layout. In our study, we perform comprehensive analysis on an HBM-like design. Subsequently, we also extend our analysis for HMC and Tezzaron designs. Figure 3 shows internal stack organizations of HBM. Each channel may be fully contained in each DRAM die in the stack. A complete set of TSVs and buffers connect each channel to the external interface.

The stacked memory consists of D data dies and E ECC dies (depending on value of D and the ECC implementation). ECC can be stored in an additional space provided by D dies or can be distributed across $D + E$ dies. Similar to ECC-DIMMs, every data request for a 512b data line also concurrently fetches its 64b ECC metadata through dedicated ECC lanes [Standard 2013]. In our paper, we use an 8-die stack with one additional ECC die for ECC or metadata information. Our organization has the same storage overhead as incurred in ECC DIMMs (12.5%).

2.4. Data Striping in 3D Memory Systems

The way data is striped in the memory system has a significant impact not only on power and performance but also reliability of the overall system. A conventional (2D) DIMM stripes a cache line across several chips. Similarly, a stacked memory system can place the cache line in one of three ways:

—**Same Bank:** Within a single bank in a single channel.
—**Across Banks:** Within a single die (channel) and striped across banks.
—**Across Channels:** Within multiple dies (channels) and striped across one bank in each channel.

2.5. Impact of Data Striping

If we use an organization that places the entire cache line in the same bank, then a failure of the bank would cause data loss of the entire cache line. To protect stacked DRAM from bank failures or channel failures, we can stripe data across banks or channels.

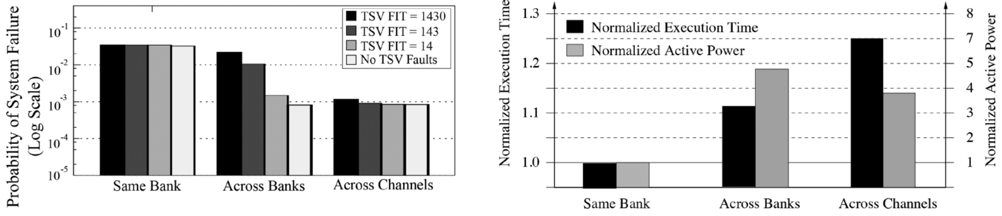

Fig. 4. Impact of data striping on Reliability, Power, and Performance. Striping data across banks or channels and using a strong 8-bit symbol-based code (similar to Chipkill) gives higher reliability. However, striping data across banks or channels comes at a significant price in performance (11%–25%) and power (3.8× to 4.7×).

Table I. Stacked Memory Failure Rates (8Gb Dies)

DRAM Die Failure Mode	Fault Rate (FIT)	
	Transient	Permanent
Single bit	113.6	148.8
Single word	11.2	2.4
Single column	2.6	10.5
Single row	0.8	32.8
Single bank	6.4	80
TSV(Complete Bank/Channel)		
TSV (Address and Data)		Sweep: 14–1,430 FIT

In such a case, each bank/channel would be responsible for only a portion of the data for the cache line, and a correction mechanism (possibly ECC scheme) can be used to fix the sub-line-granularity fault. This organization activates multiple banks/channels to satisfy each memory request and reduces bank-level parallelism. Subsequently, stacked DRAM consumes much higher power, as it activates multiple banks.

Figure 4 compares the reliability for three data mapping schemes for strong 8-bit symbol-based ECC (similar to ChipKill) for different TSV FIT rates (other parameters are described in Section 3). System failure is the occurrence of an uncorrectable fault within a seven-year lifetime. Across-Channels configuration provides the highest reliability.

Unfortunately, the reliability benefits of Across-Banks and Across-Channels come at a significant price in terms of performance and power. Figure 4 shows that striping data Across-Banks causes a slowdown of approximately 10%, and Across-Channels causes a slowdown of approximately 25%. Furthermore, Across-Channels and Across-Banks consumes 3.8× to 4.7× more active power than the Same-Bank mapping (Across-Channels takes longer to execute, consuming energy over a longer time, hence the relative reduction in power compared to Across-Banks).

Our Goal: The goal of this article is to obtain high performance and power-efficiency by maintaining the data mapping of a Same-Bank configuration while still making stacked memory robust to large granularity faults. We describe our methodology before describing our solutions.

3. EXPERIMENTAL METHODOLOGY

3.1. Fault Models and Failure Rates

Real-world field data from Sridharan and Liberty [2012] provides failure rates as Failures In Time (FIT) for DRAM chips. As TSV failure data is not publicly available, we perform a sensitivity study for TSV device FITs. We assume 0.01 to 1 device failures in 7 years (translating to Device FIT of 14 to 1,430) due to TSV faults. Table I shows

Table II. Baseline System Configuration

Processors	
Number of cores	8
Processor clock speed	3.2GHz
Last-level Cache	
L3 (shared)	8MB, 8-way, 24 cycles
Cache-line size	64B
DRAM 2×8GB 3D stacks	
Memory bus speed	800MHz (DDR3 1.6GHz)
Memory channels	8/Stack
Capacity per channel	1GB
Banks per channel	8
Row-buffer size	2KB
Data TSVs	256/Channel
Addr TSVs	24/Channel
t_{WTR}-t_{CAS}-t_{RCD}-t_{RP}-t_{RAS}	7-9-9-9-36

the failure rates per billion hours (FIT) and the failure sensitivity for our evaluations (from Nair et al. [2014]).

3.2. Simulation Infrastructure

Reliability. To evaluate reliability of different schemes, we use a industry-grade fault and repair simulator *FaultSim* [David and Prashant 2014]. We configure a scrubbing interval of 12 hours. After intervals of 12 hours, correctable transient faults are removed due to the scrubbing mechanism. We conduct the Monte Carlo simulations for $10^5$to 10^6 trials (more trails for schemes that show lower failure rates, to improve accuracy) for lifetime of 7 years and report an average.

Performance. The baseline configuration is described in Table II. The in-house system simulator uses 8 cores which share an 8MB LLC. The memory system uses 3D stacks with eight 8Gb dies for data and one additional die for ECC or metadata in the case of Citadel. Virtual-to-physical translation uses a first-touch policy with a 4KB page size.

For our evaluations, we chose all 29 benchmarks from the SPECCPU 2006 [Henning 2006] suite. We also used memory-intensive benchmarks from the PARSEC [Bienia et al. 2008] suite, such as *black*, *face*, *ferret*, *fluid*, *freq*, *stream* and *swapt*. From the BioBench [Albayraktaroglu et al. 2005] suite, we used *tigr* and *mummer*. We generated a representative slice of 1 billion instructions using Pinpoints [Patil et al. 2004].

Our evaluations execute the benchmark in rate mode, in which all eight cores execute the same benchmark. We perform timing simulation until all the benchmarks in the workload finish execution, and we measure the execution time as the average execution time of all eight cores.

Power. We measure active (read, write, refresh, and activation) power using the equations from the Micron Memory System Power Technical Note for 8Gb chip [Micron 2007, 2010]. As per HBM, the refresh interval is set to 32ms [Standard 2013; JEDEC 2011].

4. CITADEL: AN OVERVIEW

We propose *Citadel*, a robust memory architecture that can tolerate both small- and large-granularity faults effectively. Figure 5 shows an overview of Citadel. HBM provisions 64 bits of ECC for every 64 bytes, possibly in a separate ECC die [Standard 2013]. Similarly, Citadel provisions each 64B cache line with 64 bits of metadata. However, Citadel uses the ECC die to store different types of metadata information,

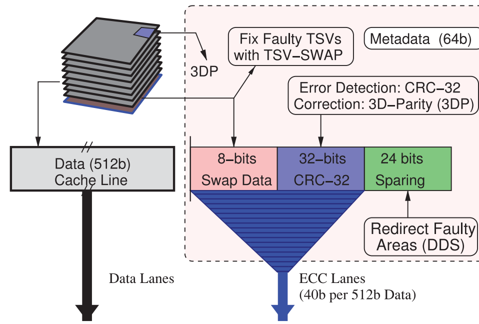

Fig. 5. Overview of Citadel.

each geared toward tolerating different types of faults. Each 64B (512b) transaction fetches 40bits of metadata over ECC lanes. The remaining 24 bits are used to provision sparing of faulty blocks. Citadel consists of three component schemes: *TSV-SWAP*, *Tri Dimensional Parity (3DP)* and *Dynamic Dual-Granularity Sparing (DDS)*.

Citadel differentiates faults in memory elements from faults in TSVs. The TSV-SWAP technique of Citadel can tolerate TSV faults by dynamically identifying the faulty TSVs and decommissioning such TSVs. The data of faulty TSVs is replicated in the metadata (up to 8 bits). TSV-SWAP protects against faulty data TSVs as well as faulty address TSVs, which tend to be even more severe in practice. Thus, TSV-Swap provides resilience to TSV faults at runtime without relying on manufacturer provided spare TSVs.

Citadel relies on CRC to detect data errors. Once an error is detected, it is corrected using the 3DP scheme, which maintains parity in three dimensions: across banks, across rows within one die, and across rows of different dies. 3DP can not only tolerate small-granularity failures such as bit and word failures but also large-granularity failures such as row and bank failures. 3DP uses one of the data banks to implement bank-level parity (storage overhead of 1.6%).

Citadel employs data sparing to avoid frequent correction of faulty data. This not only prevents the performance overheads of error correction but also makes the system more robust, as otherwise permanent faults gets accumulated over time. The DDS sparing scheme of Citadel exploits the observation that a bank either has a few small granularity faults (less than 4) or many (more than 1,000) faults; DDS spares at either a row granularity or a bank granularity. DDS uses three out of eight banks of the metadata die for sparing.

When combined, the three techniques of Citadel can tolerate TSV and multi-granularity granularity faults while consuming a storage overhead similar to an ECC DIMM (14% for Citadel versus 12.5% for ECC DIMM) and allowing the data of the cache line to be resident in the same bank. The next sections describe the three techniques in detail.

5. MITIGATING TSV FAULTS WITH TSV-SWAP

Stacked memory systems use TSVs to connect data, address, and command links between the logic die and DRAM dies. Without loss of generality, this section explains the working of TSVs, fault models, and our solution.

5.1. Background on TSV

The HBM system in this article consists of 8 channels of 256 Data TSVs (DTSV) with 24 address/command TSVs (ATSV). A memory request presents an address and

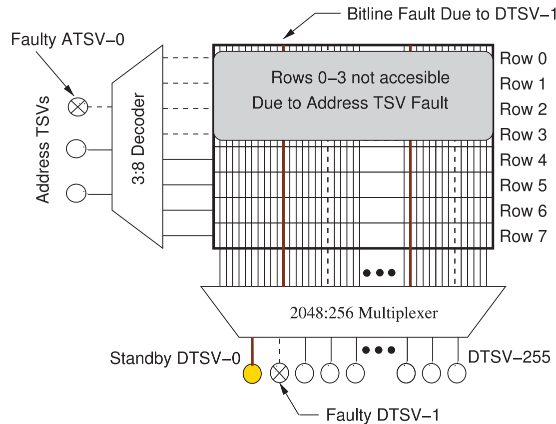

Fig. 6. Faults in Data TSV (DTSV) and Address TSV (ATSV). TSV-SWAP creates standby TSVs from existing TSVs to tolerate TSV faults (such as DTSV-1 and ATSV-0).

commands over external address/command links. Internally, TSVs transfer the address and command information for the channel to the corresponding die. For a read request for one cache line, the entire 2KB of data for the row (called a DRAM page) is addressed and brought into the sense amplifiers. From the 2KB (16Kb) page, 64B (512 bits) of data are multiplexed and transferred via the TSVs. Because there are only 256 DTSVs, each TSV will transfer data in two DDR cycles. The DRAM row (2KB) contains data for 32 cache lines. Each of these 32 cache lines is multiplexed to the same set of TSVs. Furthermore, all banks within the same die share the TSVs, which means a fault in the TSV causes multi-bank failures.

5.2. Severity of TSV Faults: DTSV vs. ATSV

The vulnerability of the system to TSV faults depends on whether the fault happens in DTSV or ATSV, as shown in Figure 6. Because the burst size for the HBM design is 2, each DTSV fault will cause 2 bits to fail in every cache line. For example, a failure of DTSV-1 will cause bit[1] and bit[257] of each cache line to fail. Faults in ATSV are even more severe; a single fault can make half of the memory unreachable, because the decoder is unable to address half of the memory space. For example, a failure of ATSV-0 makes half of the rows (Row-0 to Row-3) unreachable.

5.3. Efficient Runtime TSV Sparing with TSV-SWAP

TSV faults at manufacturing time are typically mitigated by spare TSVs provisioned for enhancing yield [Hsieh et al. 2010]. Such spare TSVs may or may not be available to the user to tolerate faulty TSVs that happen at runtime. Our proposal, *TSV-Swap,* can mitigate TSV faults at runtime without relying on manufacturer-provided spare TSVs and distinguishes between the severity of faults in address and data TSVs. TSV-SWAP differentiates between address and data TSVs with the help of a built-in test logic. Instead of relying on spare TSVs, it creates a pool of standby TSVs from the available DTSVs and uses these standby TSVs to repair the faulty DTSV and ATSV. TSV-SWAP consists of three steps, which are described as follows.

5.3.1. Creating Standby TSVs. TSV-SWAP creates stand-by TSVs by duplicating the data of predefined TSV locations into the 8-bit swap data provided by metadata in Citadel (see Figure 5). Our design designates four TSVs as standby TSVs from a pool of 256

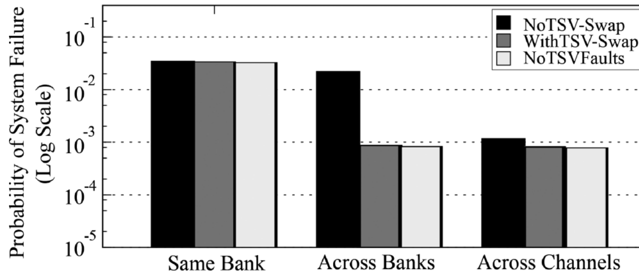

Fig. 7. TSV-SWAP is effective at mitigating TSV faults and provides almost similar performance to an ideal ChipKill system.

DTSV (DTSV-0, DTSV-64, DTSV-128, and DTSV-192). As each DTSV bursts two bits of data for each cache line, 8 bits from each cache line are replicated in the metadata (bit[0], bit[64], ... , bit[448]). The four standby TSVs that are created are used to repair any faulty TSVs that occur at runtime.

5.3.2. Detecting Faulty TSV. Citadel computes CRC-32 using address and data information. A TSV error will result in an incorrect checksum. To differentiate between TSV faults and data faults, TSV-SWAP employs two additional rows (*row1-fixed* and *row2-fixed*) per die that stores a fixed sequence of data. These rows are at locations where each bit of addresses are the inverse of each other (e.g., address *0x0000* and *0xFFFF*). On detecting a CRC mismatch, data from these fixed rows are read and compared against the predecided sequence. If there is a mismatch between the compared values, the error is highly likely (but not always) due to a TSV fault. The memory system now invokes the BIST logic, which checks for TSV faults.

5.3.3. Redirecting Faulty TSV. TSV-SWAP provisions both the DTSV and ATSV with a redirection circuit that can replace a faulty TSV with one of the standby TSVs. The redirection circuit is simply a multiplexer and a register. On detecting a TSV fault, the BIST circuitry enables the TSV redirection circuit as a corrective action against the faulty TSV. The BIST circuitry then connects one of the standby TSVs to replace the faulty DTSV or ATSV.

5.4. Result: TSV-SWAP with ChipKill

We analyze the effectiveness of TSV-Swap at mitigating TSV faults for a system employing ChipKill. Unfortunately, the FIT rate data for TSV faults is not available publicly, so for this section, we assume a high TSV fault rate (1,430 FIT, corresponding to one TSV-caused die failure every 7 years) to assess the effectiveness of TSV-Swap at high TSV fault rate. Figure 7 shows the probability of system failure for the three configurations (No TSV-Swap, With TSV-Swap, and No TSV Faults) for the three data mappings. For all systems, TSV-SWAP achieves a resilience similar to that of not having any TSV faults, even with the assumed high failure rate for TSVs. We conclude that TSV-SWAP is highly effective at mitigating TSV failures.

5.5. TSV-SWAP for Alternate Stacked Memory Organizations

Until now we have evaluated our design of TSV SWAP for an HBM-like organization. However, stacked memories can have alternate organizations in the placement of TSVs. Figure 8 shows two alternate organizations of stacked memory systems that reorganize channels and banks by changing the placement of TSVs.

The first organization of stacked memory, Organization-A, is an HMC-like organization. In Organization-A, the channel(s) are organized vertically across dies. Every

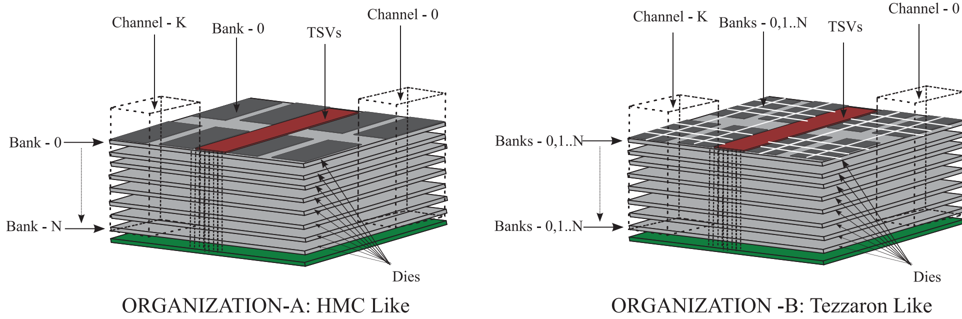

Fig. 8. Alternate 3D stacked memory organizations. Organization-A has channels across dies (vertically) and all banks in this channel are in different dies and is similar to an HMC system. Organization-B has channels across dies (vertically) and is similar to a Tezzaron stacked memory system. In this case, each die has a portion of all banks in that channel.

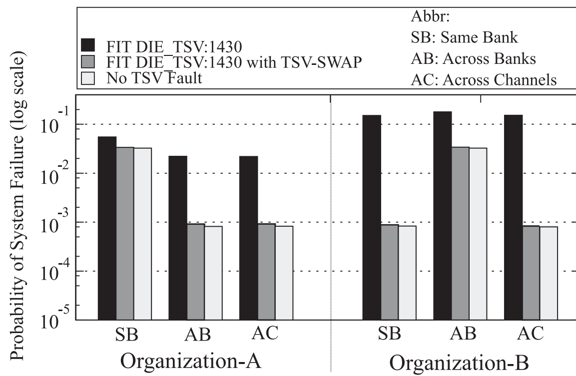

Fig. 9. Organization-A (HMC-like) and Organization-B (Tezzaron-like) can be more sensitive to TSV faults when compared to an HBM-like organization. TSV Swap mitigates almost all TSV faults.

die contributes a single bank to each channel. The TSVs are distributed across dies and every channel in every die requires individual buffers. The second organization, Organization-B, of stacked memory is a Tezzaron-like organization. In Organization-B, the channel(s) that are organized vertically across dies. However, every die holds a portion of multiple banks for a channel. Because of this, the number of address and data TSVs per channel per die increases.

Figure 9 shows the probability of system failure for two alternate stacked memory configurations. Our evaluations show that Tezzaron-like designs are more prone to TSV faults because higher density of TSVs for data and address. As every physical bank is further divided into several logical banks, placing data across these logical banks has the same effect as placing data in the same bank. Because of this, the across bank data placement has the lowest reliability in a Tezzaron-like design. An HMC-like design has similar trend to that of a HBM-like configuration. Even for alternate organizations, TSV-SWAP achieves a resilience similar to that of not having any TSV faults.

5.6. Reducing the Complexity of TSV-SWAP

Architecting any Data TSV to swap between address, command and other data TSVs increases the complexity of the swap logic. To overcome this, TSV-SWAP uses a set structure for swapping TSVs. In this structure, a set of TSVs (address/control+data) co-located with a fixed Standby Data-TSV (S-TSV). Only one TSV encountering a fault

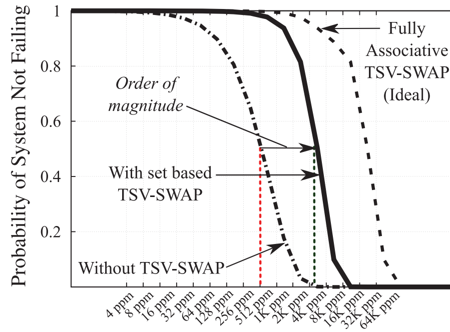

Fig. 10. Set-based TSV-SWAP can handle 10× more TSV failures before it causes system failure. In our analysis, a set consists of 70 TSVs, 63 data+*1 Data Swap-TSV*+6 address/command TSVs.

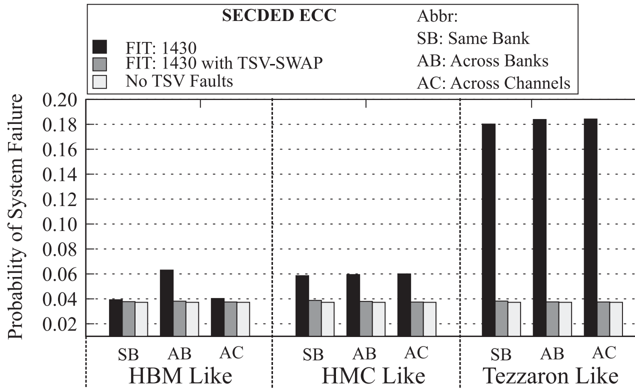

Fig. 11. TSV-SWAP is effective at mitigating TSV faults and provides almost similar performance to an ideal system employing SECDED codes.

can be swapped with its *S-TSV* in a set. In our analysis, a set consists of 70 TSVs, 63 data+*1 Data Swap-TSV*+6 address/command TSVs. We perform bucket and balls analysis to determine the probability of system failure for such set group. Figure 10 shows that such set-based TSV-SWAP can handle 10× more TSV failures when compared to a system that does not employ TSV-SWAP. An ideal fully associative (complex) TSV-SWAP circuitry provides another 10× higher reliability when compared to the SET-based scheme.

6. EFFECTIVENESS OF TSV SWAP FOR MEMORY SYSTEM EMPLOYING SINGLE ERROR CORRECTION AND DOUBLE ERROR DETECTION (SECDED)

Until now, we have assumed a system that employs a symbol-based error correcting code like ChipKill. These symbol-based codes can correct large-granularity faults and single-bit errors. Fortunately single or multiple random-bit errors can be corrected using Bose-Chaudhuri-Hocquenghem (BCH) codes, including Hamming Codes [Lin and Costello 2004]. Hamming codes have a bit storage overhead of log_2*(Size of the Code Word)+1* (including additional error detection). They provide single error correction, double error detection (SECDED) computed over an 8B codeword requires 8 additional bits for every 64 bits. Decoding and encoding complexity, check bit overhead and latency increase with the strength of the ECC.

Figure 11 shows the effectiveness of TSV SWAP for a system that employs SECDED-based ECC. SECDED provides lower reliability when compared to ChipKill; however,

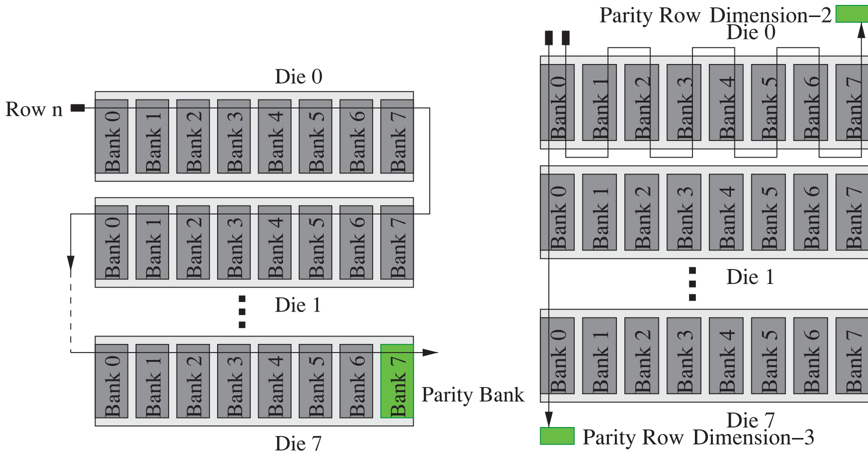

Fig. 12. Dimension 1 stripes parity across a single row in every bank for all dies and generates a row in the parity bank. Dimension 2 stripes parity across all row in every bank within a die and generates a parity row. Dimension 3 stripes parity across all rows in single bank across dies and generates a parity row.

TSV SWAP enables SECDED to overcome errors due to TSV faults and provides reliability close to an ideal system that employs SECDED protection.

Furthermore, Figure 11 shows that Tezzaron-like designs are more vulnerable to TSV faults because of their higher density of TSVs. Furthermore, since TSVs may cause large-granularity failures, SECDED is ineffective in mitigating them. Subsequent sections assume that the system employs symbol-based ECC code (ChipKill Like) and TSV faults are mitigated with TSV SWAP.

7. TRI-DIMENSIONAL PARITY (3DP)

The second component of Citadel targets efficient error detection and error correction of data values. Several error detection codes such as SECDED, Checksums, and CRC-32 or CRC64 are used in commercial systems [Peterson and Brown 1961; Koopman 2002; Li et al. 2011]. We found that CRC-32 has reasonable detection coverage and storage efficiency. Citadel provisions each line with a 32-bit cyclic redundancy code (CRC-32), which is highly effective[1] at detecting data errors [Peterson and Brown 1961; Sim et al. 2013]. Citadel uses a novel scheme called *Tri-Dimensional Parity (3DP)* to correct data errors at multiple granularities. In 3DP, even if one dimension encounters two faults, they are highly unlikely to fall into the same block in the other two dimensions. On detecting an error, the memory contents are read and the error gets corrected using parity.[2]

7.1. Design of Dimension 1

Figure 12 shows the design of Dimension 1. It computes the parity for a row in every bank across dies as specified in Equation (1). This requires dedicating a range of single bank addresses as a parity bank for the entire stack (1.6% overhead, for our 8-channel

[1]The probability of overlapping CRC-32 checksum is $\frac{1}{2^{32}} \approx 10^{-10}$. For false negative, the failed element should have an overlapped CRC-32. The probability that an element fails is less than 10^{-6}. Thus, the effective probability of an overlapping CRC-32 is negligibly small ($\ll 10^{-16}$).

[2]Error correction may take 700ms; however, given that correction is invoked once every few months, this results in negligible performance overheads. We discuss a scheme to avoid persistent errors in the next section.

system, with 8 banks for each channel).[3] A parity bank helps mitigate single-bank faults. However, a one-dimensional parity (1DP) scheme is intolerant of multiple faults. Even if a single-bit failure occurs after a single-bank failure, it results in data loss.

$$ParityBank[row_n] = Die_0.Bank_0[row_n] \oplus Die_0.Bank_1[row_n] \oplus \cdots \oplus Die_7.Bank_6[row_n] \quad (1)$$

7.2. Design of Dimensions 2 and 3

Figure 12 shows the design of Dimensions 2 and 3. In Dimension 2, parity is taken across all rows in all banks within a die. Equation (2) shows the computation *Parity Row* in Dimension 2 for Die 0. Because there are 9 dies (including the metadata die), the storage overhead is 9× the size of a DRAM row for each dimension.

$$ParityRowDim2_{Die0} = [Bank_0[row_0] \oplus Bank_0[row_1] \oplus \cdots \oplus Bank_7[row_n]]_{Die0} \quad (2)$$

Dimension 3 computes parity across dies for all rows in a single bank. Equation (3) shows the computation for *Parity Row* in Dimension 3 for Bank 0. Because there are 8 banks per die, the storage overhead of is 8×size of DRAM row. While Dimension 1 is designed to tolerate bank failures, Dimensions 2 and 3 prevent independent row, word, and bit failures. When used together, 3DP can correct multiple errors that occur at the same time within a stack.

$$ParityRowDim3_{Bank0} = [Die_0[row_0] \oplus Die_0[row_1] \oplus \cdots \oplus Die_7[row_n]]_{Bank0} \quad (3)$$

7.3. Reducing Overheads for Parity Update

We avoid the performance overheads of updating the parity for Dimensions 2 and 3 by keeping the parity information on-chip. The size of the row buffer of the stacked DRAM we simulate is 2KB [Consortium 2013; Standard 2013]. Thus, maintaining Dimensions 2 and 3 would require a storage overhead of 34KB (9 rows for Dimension 2 and 8 rows for Dimension 3), which can be kept at the memory controller. Thus, updating the parity for Dimensions 2 and 3 can be done on-chip with negligible timing and power overheads.

The total size of parity for Dimension 1 is equal to 1Gb (128MB), which would be impractical to duplicate at the memory controller side. To reduce the parity update overheads for Dimension 1, we employ parity caching within the on-chip LLC. For Dimension 1, every parity cache line is responsible for 63 data lines from 63 different banks. Thus, we expect accesses to parity lines to have very high temporal locality. Figure 13 shows the operation of a system that implements on-demand parity caching within the LLC for a writeback request to a data line (action ❶).

To update the parity information, we need to get the XOR of the old data and new data of the line for which a writeback request is being made. The memory controller performs such a Read Before Write (RBW) request to obtain the old information of the line (action ❷). As the row was recently opened, RBW tends to be a row-buffer hit. The XOR forms a parity update. The memory controller then checks the LLC for the parity line associated for the address for which writeback is being made. In the

[3]Parity bank is an abstraction, and such a bank can have addresses across multiple physical banks in a stack. This can be done by swapping 2 bits (1 lower bank bit and 1 higher channel bit) while addressing the parity bank. This prevents one physical bank from becoming a bottleneck.

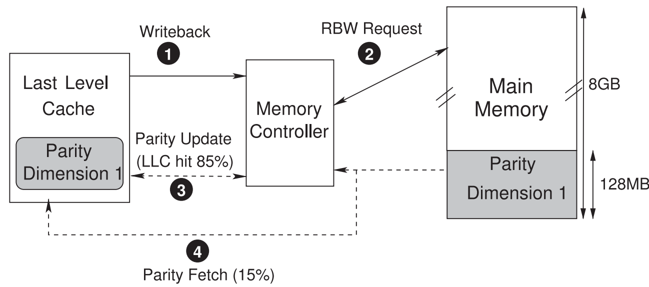

Fig. 13. Memory system employing on-demand parity caching for Dimension 1 within the LLC (figure not to scale).

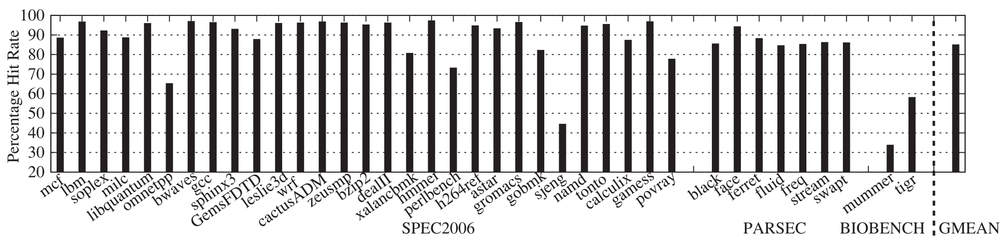

Fig. 14. Hit rate for parity caching of Dimension 1.

common case (85% of the time, on average) the parity line is found in the LLC and the parity is updated with the XOR value (action ❸). In the uncommon case that the parity information for Dimension 1 is not found in the LLC, then parity information is fetched from the memory (action ❹), installed in the LLC, and the parity information is updated.

Figure 14 shows the LLC hit-rate for parity update requests. On average, the hit rate is 85%, showing that parity caching is quite effective. The BIOBENCH workloads mostly perform read operations, with writes sparsely distribute between a large number of writes. Hence, read requests tend to evict parity lines. However, since the frequency of writes for BIOBENCH is less, a low hit rate for parity update results in negligible performance loss.

7.4. Error Detection and Correction Using 3DP

On every read request, 3DP works in two phases. The first phase consists of fast error checking using CRC-32 code. For most requests, this phase will report no errors. However, in the rare case of a reported error (once in a few months), the second phase is activated and the whole memory is read. 3DP then isolates the fault(s) using all three dimensions of parity across the stack. If it is a small granularity bit, word, or row fault, then Dimensions 2 and 3 parity can fix such errors. However, large-granularity faults, such as column and bank faults, are corrected using Dimension 1 parity. In the event of simultaneous multi-granularity faults, Dimensions 2 and 3 parity help isolate small-granularity faults and Dimension 1 parity helps isolate the large-granularity fault.

7.5. Results for 3DP

The 3DP scheme allows the memory system to retain the cache line within the same bank, and yet be able to correct bit, word, row, column, and bank failures. We compare the resilience, performance, and power of the 3DP scheme to a theoretical scheme that

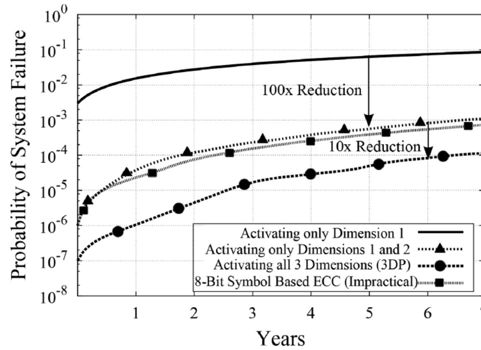

Fig. 15. 3DP has 7× more resilience than an 8-bit symbol-based ECC code for tolerating large-granularity failures in stacked memory. 3DP has 10× more resilience than 2DP.

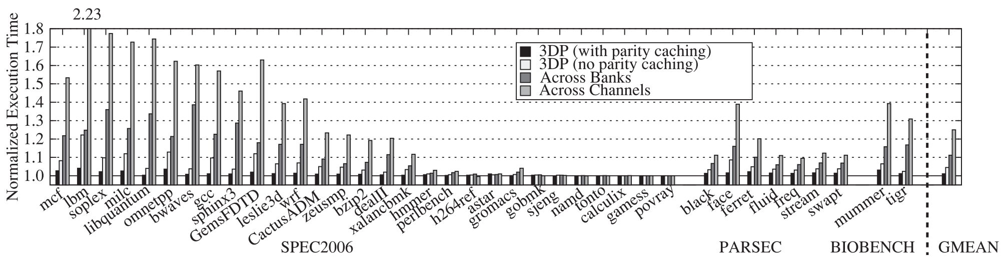

Fig. 16. Normalized execution time: 3DP has negligible slowdown, whereas data striping causes 10%–25% slowdown.

employs an 8-bit symbol-based coding with data striping. For a fair comparison between the two schemes, we assume that TSV-Swap is enabled for both the 8-bit symbol-based code and 3DP.

7.5.1. Resilience. Figure 15 compares the multi-dimensional parity scheme with a very strong 8-bit symbol-correcting code striped across channels. Enabling only a single dimension of parity (at Bank Level) does not improve resilience against multiple faults that occur concurrently. A single-dimensional parity scheme is unable to correct these faults. By enabling all three dimensions, 3DP achieves a 1,000× improvement in resilience. Furthermore, 3DP achieves 7× stronger resilience than an 8-bit symbol-based ECC because it can handle higher number of multiple concurrent faults.

7.5.2. Performance. Figure 16 compares the execution time of 3DP to the organizations that stripe data either across a bank or a channel. The execution time is normalized to a baseline that retains the cache line within the same bank and pays no overhead for error correction. The 3DP scheme with caching has performance within 1% of the baseline, and 3DP without caching degrades performance by 4.5%. Thus, parity caching is highly effective at mitigating the performance impact of parity updates. Alternative schemes, that rely on striping the data in different banks or channels, degrade performance by as much as 10% to 25%, on average, due to the loss of bank-/channel-level parallelism. Thus, 3DP not only improves the resilience of stacked memory compared to data striping but also helps brings the performance impact of fault tolerance to a negligible level.

7.5.3. Power. Accessing multiple banks or channels to satisfy every memory request also has the disadvantage that it consumes significantly higher power. Our proposed

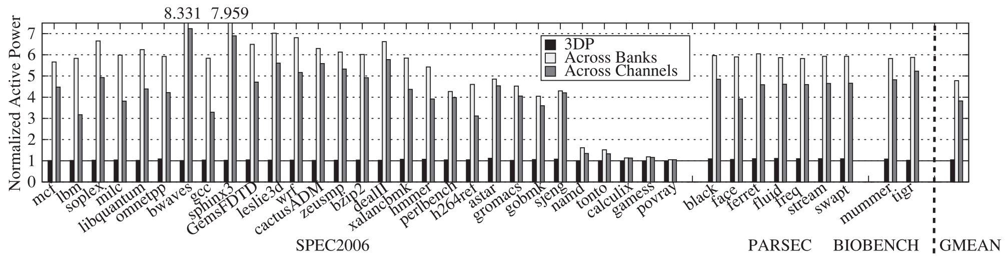

Fig. 17. Active power consumption: 3DP has negligible power overheads, whereas data striping has 3× to 5× greater overhead.

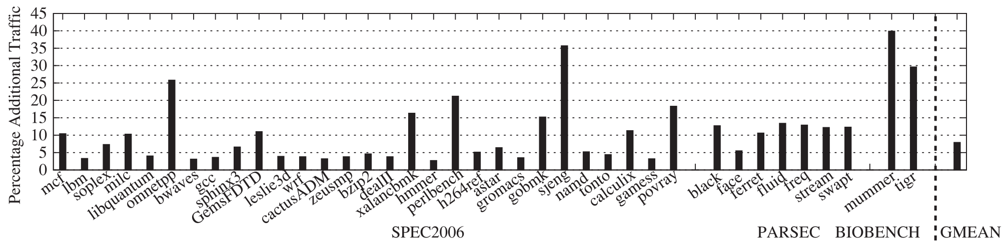

Fig. 18. Percentage of additional memory traffic: 3DP with Dimension 1 caching, on an average incurs only 8% with a maximum of 40% for workloads with low LLC dimension parity hit rate.

3DP design allows Citadel to place the entire cache line in one bank and thus activate only one bank per read request. This not only reduces the activation power but also improves memory-level parallelism, compared to the Across-Bank and Across-Channel configuration. Figure 17 shows the active power for 3DP, Across-Bank, and Across-Channel configuration, normalized to the fault-free baseline that places the cache line in the same bank. On average, 3DP increases active power by only 4%, whereas Across-Bank and Across-Channel configurations increase active power by almost 3× to 5× of higher bank/channel activations and row conflicts.

7.5.4. Additional Memory Traffic. 3DP updates Dimension 1 parity for every write and accesses this from memory. Because of this, there is additional traffic on every write. To overcome this, 3DP uses parity caching of Dimension 1 parity. Figure 18 shows the additional traffic after caching Dimension 1 parity. On average, dimension-1 caching helps in reducing the average additional memory traffic to 8%. The additional memory traffic is correlated to the hit rate of Dimension 1 parity in last-level cache. For instance, omnetpp and sjeng have low Dimension 1 parity hit rates and, therefore, have up to 35% higher traffic. Since writes in BIOBENCH are sparsely distributed, additional memory traffic does not have a significant impact on its performance.

8. DYNAMIC DUAL-GRANULARITY SPARING (DDS)

The 3DP scheme performs error correction by recomputing the data based on parity information. However, this can be a time-consuming process (recomputing parity and isolating the fault in each dimension). Fortunately, faults do not occur frequently, so employing a slow correction mechanism is a viable option. However, if the faults are permanent, then the correction scheme will be invoked frequently and cause unacceptable performance degradation. Citadel avoids this by using dynamic sparing, whereby a data item once corrected is redirected to an alternate location. The key question in designing a data-sparing scheme is the granularity of sparing. Sparing at row granularity would be storage efficient; however, it would be fairly complex to tolerate bank

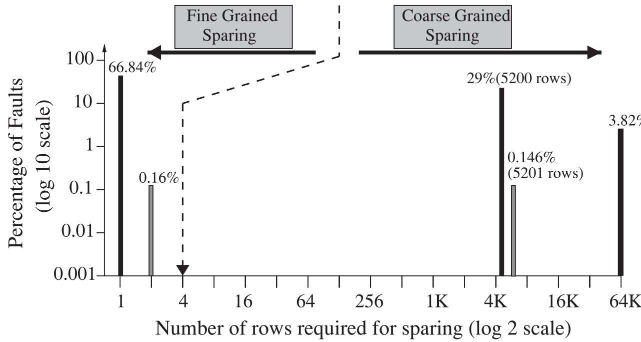

Fig. 19. Permanent fault affects either very few (<4) rows or large number of (>1,000) rows.

Table III. Number of Failed Banks (for a System with ≥1 Bank Fail)

Num Faulty Banks	1	2	3+
Probability	66.98%	32.98%	0.04%

failures, as the redirection structures associated with row sparing would require several tens of thousands of entries. We can implement sparing at a bank granularity but suffer significant underutilization of spare area. Thus, uniform sparing is either complex or inefficient. To address this dichotomy, Citadel is provisioned with *Dynamic Dual-granularity Sparing (DDS)*. We present the key observation that motivates DDS.

8.1. Key Observation: Failures Tend to be Bimodal

Only for the analysis in this section, we will classify all faults that are smaller than or equal to a row fault as causing a row failure. These faults will consume one entry for a row-sparing architecture. A large-granularity fault would consume many entries of row sparing. Figure 19 shows the distribution of the number of rows that are used by a faulty bank, on average, based on Monte Carlo simulations using FaultSim [David and Prashant 2014].

The number of failures show a bimodal distribution. The smaller-granularity faults do not occur in many multiples. In fact, in all our simulations, no more than two rows per bank were affected by a small-granularity fault within a scrubbing interval. However, there are two peaks; one at 5,200 rows (most likely due to sub-arrays) and another at 65K rows (size of a bank). A row-sparing architecture would be not effective at tolerating 65K spare rows for a failed bank, because the sparing-associated table would become impractically large to build and search on every access. Therefore, DDS implements two granularities of sparing: either a row or a bank.

8.2. Budgeting Spare Rows and Spare Banks

DDS partitions faults into small- and large-granularity faults and then replaces small-granularity faults with rows and large-granularity faults with a bank. Based on the data shown in Figure 19, we deem any bank having more than four faulty rows as a bank failure and spare that bank. Given that a bank can have at most four row failures before the bank gets spared, the number of spare rows required would be equal to four times the number of banks (64 banks will have 256 spare rows).

The number of spare banks depends on the bank failure rate. Table III shows the distribution of faulty banks for a system that has at least one failed bank (more than four row faults), derived using Monte Carlo simulations with FaultSim [David and

Prashant 2014]. Even under our conservative definition of bank failure, we need at most two spare banks to handle 99.96% of the systems that have a bank failure, so we employ two spare banks in our design.

8.3. Design of Dynamic Dual-Granularity Sparing

DDS has two components;: the spare area and the redirection table. Because we employ two granularities of sparing, we have two redirection tables: one at row granularity and the other one at a bank granularity.

8.3.1. Spare Area. The metadata die in *Citadel* has eight banks. TSV and 3DP use five banks within the metadata die for storing CRC-32 and TSV-SWAP–related information. DDS uses the three remaining banks for sparing. These three banks are partitioned into coarse-granularity sparing banks (*spare bank-0* and *spare bank-1*) and a fine-granularity bank (*spare bank-2*) for row-based sparing.

8.3.2. Row Remap Table (RRT). DDS uses RRT to associate faulty row addresses with spare row addresses. Each RRT entry contains a valid bit (1), the source row ID (16 bits), and a destination row ID (16 bits). Each fault is tagged with a faulty row address and its corresponding spare address. Because DDS supports at most four spare rows for each bank, each bank has four entries in RRT. The overhead of RRT for our 8-die (eight banks per die) system is approximately 1KB and the RRT is stored on-chip. A memory access will check the four RRT entries of the given bank for a valid row ID match. On a valid match, the spare row is accessed.

8.3.3. Bank Remap Table (BRT). If all four spare rows dedicated to a bank get exhausted, and a new fault appears, then the fault is treated like a large-granularity (bank) failure and coarse-granularity sparing is invoked. The data from the failed bank is repaired and relocated to the spare bank. A two-entry Bank Remap Table (BRT) provides redirection for faulty banks. Each BRT entry contains a valid bit, the ID of the failed bank (6-bit ID), and ID of the spare bank (1-bit spare bank ID, to select one of two spare banks). Prior to looking up the RRT, the BRT is located on-chip and is probed on every memory access for a match. On a BRT hit, the spare bank is accessed.

9. SINGLE ERROR CORRECTION (SEC) TO MITIGATE CORRECTION LATENCY

Several studies have shown that soft errors (α particle strikes), scaling errors, and retention errors are usually manifested as single bit errors. Unfortunately, Citadel, in a worst case, can take up to 700ms to correct such errors. We propose using Single Error Correction (SEC) to optimize Citadel for the common case of single bit errors. Single Bit Error Correction using Hamming Code for a 512-bit cache line requires an additional 10 bits.

We propose two techniques to implement SEC in Citadel without using additional area:

—First, we do not use an additional bank for sparing small granularity failures. We propose using the LLC to store values from these failed rows persistently. This will reduce the capacity of LLC by only 256KB (3.3% for an 8MB LLC).
—Second, since SEC uses 10 bits, we need space to store these additional 10 bits. To do this, we employ the recently unused additional bank (previously used to spare small granularity faults) to store 8 bits per cache line (1 bank every 64 banks). To accomodate additional two bits, we downgrade CRC-32 (32bits) into CRC-30 (30 bits).[4] We use these additional bits to store the 9th and 10th bits for SEC.

[4]CRC-30 is already used in CDMA technology.

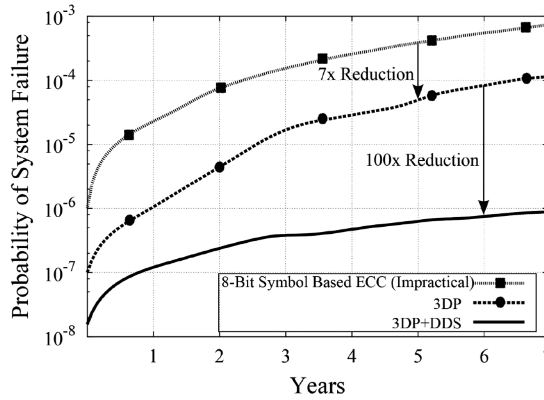

Fig. 20. Resilience: 3DP+DDS provid es 700× more resilience than symbol-based codes.

9.1. Quantifying the Effect of Using CRC-30 Checksum Against a CRC-32 Checksum

The chance of a CRC-32 checksum overlapping is $\frac{1}{2^{32}}(\approx 10^{-10})$. The baseline design uses CRC-32 to maximize the bandwidth used on the ECC lanes. In the SEC-based optimization for soft errors, the CRC-30 checksum will have an overlapping probability of $\frac{1}{2^{30}}(\approx 10^{-9})$. Since the probability that an element fails is much less than 10^{-6}, the CRC-30 checksum has a detection probability of $\ll 10^{-15}$, as compared to CRC-32 checksum with a detection probability of $\ll 10^{-16}$.

9.2. Operation

For every access, we will update these 10 bits every cache line using the ECC lanes. SEC-based correction works in three steps. On detecting a CRC error, the stacked memory system uses SEC to correct errors. Unfortunately, in case of multi-bit errors, SEC may not be able to correct the error. To avoid this, on correcting an error using SEC, Citadel recomputes the CRC again. In the common case of single-bit errors, this will usually result in a CRC match. On a CRC match, Citadel infers that the error is corrected. In case of a CRC mismatch, Citadel denotes this as a multi-bit error and employs the longer latency error correcting of 3DP.

10. OVERALL RESULTS

This section explains the impact of tying together TSV Swap, 3DP, and DDS and explains the overheads in implementing Citadel.

10.1. Tying It Together

Figure 20 compares the effectiveness of 3DP with DDS to an 8-bit symbol correcting code. For all systems, we assume that TSV-SWAP is enabled. When applied with 3DP, DDS delivers a 700× improvement in resilience compared to the baseline strong 8-bit symbol-based ECC code. DDS removes 99.995% of all transient faults and 99.996% of all the permanent faults with a 12-hour scrubbing interval and thus prevents the accumulation of faults. Therefore, DDS can protect against multiple faults if they occur during different scrub intervals. Overall, these results show that Citadel can provide a reliability improvement of almost 3 orders of magnitude. It does so without requiring the system to stripe data for a cache line across banks.

10.2. Storage Overhead of Citadel

Citadel relies on having an extra die for storing metadata for the eight data dies (12.5% overhead). In addition, bank-level parity requires dedicating one of the data bank for storing parity (1.6% overhead, 1 bank out of 64 banks). For 3DP, we keep parity for Dimensions 2 and 3 on-chip (34KB overhead), and the redirection tables of DDS incur about 1KB overhead, for a total SRAM overhead of only 35KB. Thus, Citadel provides 700× better reliability while requiring a storage overhead of 14%, which is similar to the overhead of ECC DIMM (12.5%).

11. RELATED WORK

Memory reliability for emerging memory technologies and existing DRAM systems has become an important topic. We describe the schemes that are most relevant to our proposal.

Citadel employs TSV-SWAP to mitigate faulty TSVs. Faulty TSVs can be avoided at manufacturing time using spare TSVs. Several techniques have been proposed for "swapping in" such redundant TSVs to replace faulty TSVs in a 3D die stack [Jiang et al. 2012]. To the best of our knowledge, this article is the first to address runtime mitigation of TSVs and without relying on manufacturer-provided spare TSVs.

Two prior works are closely related to our work. The first prior work proposes techniques to reliably architect stacked DRAM caches [Sim et al. 2013]. It uses CRC-32 to detect errors in caches. However, correction is performed simply by disabling clean lines and replicating dirty lines. While such correction can be useful for caches, disabling random locations of lines is an impractical option for main memory. Furthermore, replicating all the data for main memory leads to a capacity loss of 50% and doubles the memory activity. Our work provides low-cost and effective fault tolerance for using stacked DRAM as main memory. The second prior work proposes a low overhead ChipKill code that can be used with current ECC-DIMMs without using additional DIMMs [Jian et al. 2013]. This work also uses a combination of error detection and correction codes but does not talk about TSV failures and efficient dual grain sparing.

Yoon and Erez [2010] proposed *Virtual and Flexible ECC*. Rather than using uniform error correction across the entire memory space, it allows the user to specify stronger levels of ECC for high-priority applications and weaker levels of ECC for low-priority applications. Citadel uses multi-dimensional parity rather than multi-tiered ECC. Citadel is more area-efficient and does not require any support from the OS.

Efficient memory repair for bit-level faults has been proposed for both SRAM [Roberts et al. 2007; Wilkerson et al. 2008] and DRAM [Nair et al. 2013]. However, such techniques are effective only for random bit errors and cannot tolerate large-granularity faults. Erasure Codes can identify faulty chips to be disabled [Nerl et al. 2007, 2008; Yoon et al. 2012]. However, they operate only at one granularity. Unlike erasure codes, DDS enables flexible granularity sparing.

Citadel uses parity for error correction, as do other schemes, such as RAID [Thomasian and Menon 1997]. BCH codes can be used to provide protection for multiple-bit errors (e.g., 6 or more bits) [Li et al. 2011; Wilkerson et al. 2010]. Unfortunately, strong BCH codes cannot handle large-granularity faults without significant overheads. Figure 21 compares the resilience of Citadel with a strong ECC scheme (6EC7ED) and with RAID-5. Because these schemes are not resilient to TSV faults, we assume a memory system with no TSV faults. Even after discounting for TSV faults, these schemes end up having orders of magnitude higher failure rates than Citadel. A RAID-5 scheme provides 89× improvement in resilience compared to 6EC7ED. Citadel provides 1,000× more resilience than a RAID-5 scheme.

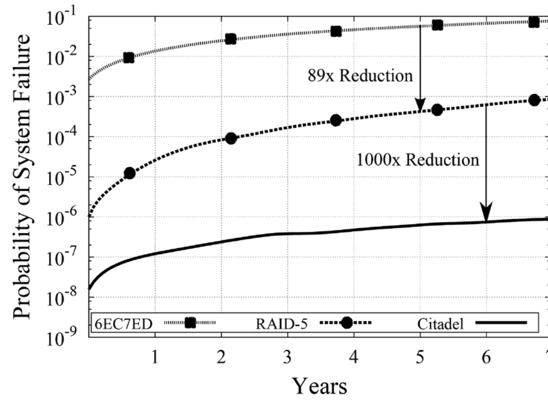

Fig. 21. Comparing resilience of Citadel to strong ECC codes (6EC7ED) and RAID-5.

12. CONCLUSION

Memory stacking introduces new multi-bit failure modes, exacerbating the large-granularity faults identified by DRAM field studies. Typical approaches tolerate only random-bit failures and tolerating large-granularity failures (such as tolerating chip failures using ChipKill) typically relies on striping data to multiple chips. Transposing such data striping to stacked memory systems causes significant slowdown and 3× to 5× power overheads. This article proposes Citadel to tolerate large-granularity faults efficiently and makes the following contributions:

(1) TSV-SWAP, which mitigates TSV faults at runtime, without relying on manufacturer-provided spare TSVs. It remains effective even at high TSV failure rates.
(2) Tri-Dimensional Parity (3DP), which can correct a wide variety of multi-granularity faults.
(3) Dynamic Dual-granularity sparing (DDS), which can spare faulty data blocks either at a row granularity or at a bank granularity to avoid the accumulation of permanent faults and frequent error correction.

Our evaluations with real-world fault data for DRAM chips shows that combining these three schemes is highly effective for tolerating high rate of TSV failures and memory failures. We show that 3DP improves reliability of stacked memory by 7×, and when combined with DDS by 700×, compared to a symbol-based code that stripes data across banks or channels. Citadel provides high reliability while maintaining high performance and low power, requiring a storage overhead close to ECC DIMMs (14% vs. 12.5%).

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their valuable comments and feedback. We also thank Vilas Sridharan for his comments on DRAM scaling and FIT rates. We are also grateful all members of our research group, CARET Lab at Georgia Tech and AMD Research for their insightful feedback.

REFERENCES

K. Albayraktaroglu, A. Jaleel, null Xue Wu, M. Franklin, B. Jacob, null Chau-Wen Tseng, and D. Yeung. 2005. BioBench: A benchmark suite of bioinformatics applications. In *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS'05)*. 2–9.

Christian Bienia, Sanjeev Kumar, Jaswinder Pal Singh, and Kai Li. 2008. *The PARSEC Benchmark Suite: Characterization and Architectural Implications*. Technical Report TR-811-08. Princeton University.

Jay Bolaria. 2011. Micron reinvents DRAM memory. In *Microprocessor Report (MPR)*.

Hybrid Memory Cube Consortium. 2013. Hybrid Memory Cube Specification 1.0. Retrieved from hybridmemorycube.org.

Roberts David and Nair Prashant. 2014. FaultSim: A fast, configurable memory-resilience simulator. In *The Memory Forum: In conjunction with ISCA-41*.

Timothy J. Dell. 1997. *A White Paper on the Benefits of ChipkillCorrect ECC for PC Server Main Memory*. Technical Report 11/19/97. IBM.

Manek Dubash. 2004. Not hot swap but "fail in place." In *TechWorld*. Retrieved from http://features.techworld.com/storage/960/not-hot-swap-but-fail-in-place/.

John L. Henning. 2006. SPEC CPU2006 benchmark descriptions. *SIGARCH Comput. Archit. News* 34, 4 (Sept. 2006), 1–17. DOI: http://dx.doi.org/10.1145/1186736.1186737

Tim Hollis. 2012. Modeling and simulation challenges in 3D memories. In *DesignCon*.

Ang-Chih Hsieh, TingTing Hwang, Ming-Tung Chang, Min-Hsiu Tsai, Chih-Mou Tseng, and Hung-Chun Li. 2010. TSV redundancy: Architecture and design issues in 3D IC. In *Proceedings of the Design, Automation Test in Europe Conference Exhibition (DATE), 2010*. 166–171. DOI: http://dx.doi.org/10.1109/DATE.2010.5457218

JEDEC. 2011. JS Choi DDR4 Mini-Workshop. Retrieved from http://jedec.org/sites/default/files/JS Choi DDR4 miniWorkshop.pdf.

Xun Jian, Henry Duwe, John Sartori, Vilas Sridharan, and Rakesh Kumar. 2013. Low-power, low-storage-overhead chipkill correct via multi-line error correction. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC'13)*. ACM, New York, NY, Article 24, 12 pages. DOI: http://dx.doi.org/10.1145/2503210.2503243

Li Jiang, Qiang Xu, and B. Eklow. 2012. On effective TSV repair for 3D-stacked ICs. In *Proceedings of the Design, Automation Test in Europe Conference Exhibition (DATE)*. 793–798.

Uksong Kang, Hoe-Ju Chung, Seongmoo Heo, Soon-Hong Ahn, Hoon Lee, Soo-Ho Cha, et al. 2009. 8Gb 3D DDR3 DRAM using through-silicon-via technology. In *Proceedings of the 2009 IEEE International Solid-State Circuits Conference - Digest of Technical Papers (ISSCC'09)*. 130–131,131a. DOI: http://dx.doi.org/10.1109/ISSCC.2009.4977342

Jung-Sik Kim, Chi Sung Oh, Hocheol Lee, Donghyuk Lee, Hyong-Ryol Hwang, Sooman Hwang, et al. 2011. A 1.2V 12.8GB/s 2Gb mobile Wide-I/O DRAM with 4x128 I/Os using TSV-based stacking. In *Proceedings of the 2011 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC'11)*. 496–498. DOI: http://dx.doi.org/10.1109/ISSCC.2011.5746413

P. Koopman. 2002. 32-bit cyclic redundancy codes for internet applications. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN'02)*. 459–468. DOI: http://dx.doi.org/10.1109/DSN.2002.1028931

Sheng Li, Ke Chen, Ming-Yu Hsieh, N. Muralimanohar, C. D. Kersey, J. B. Brockman, A. F. Rodrigues, and N. P. Jouppi. 2011. System implications of memory reliability in exascale computing. In *Proceedings of the 2011 International Conference on High Performance Computing, Networking, Storage and Analysis (SC'11),* 1–12.

S. Lin and D. J. Costello, Jr. 2004. *Error Control Coding: Fundamentals and Applications*. Prentice Hall, Englewood Cliffs, NJ.

Micron 2007. *Calculating Memory System Power for DDR3*. Micron.

Micron 2010. *MT41J512M4:8Gb QuadDie DDR3 SDRAM Rev. A 03/11*. Micron.

Prashant J. Nair, Dae-Hyun Kim, and Moinuddin K. Qureshi. 2013. ArchShield: Architectural framework for assisting DRAM scaling by tolerating high error rates. In *Proceedings of the 40th Annual International Symposium on Computer Architecture (ISCA'13)*. ACM, New York, NY, 72–83. DOI: http://dx.doi.org/10.1145/2485922.2485929

Prashant J. Nair, David A. Roberts, and Moinuddin K. Qureshi. 2014. Citadel: Efficiently protecting stacked memory from large granularity failures. In *Proceedings of the 2014 47th Annual IEEE/ACM International Symposium on Microarchitecture*. 51–62. DOI: http://dx.doi.org/10.1109/MICRO.2014.57

John Nerl, Ken Pomaranski, Gary Gostin, Andrew Walton, and David Soper. 2007. System and method for applying error correction code (ECC) erasure mode and clearing recorded information from a page deallocation table. In *US 7313749 B2 - Patent*.

John Nerl, Ken Pomaranski, Gary Gostin, Andrew Walton, and David Soper. 2008. System and method for controlling application of an error correction code (ECC) algorithm in a memory subsystem. In *US 7437651 B2 - Patent*.

H. Patil, R. Cohn, M. Charney, R. Kapoor, A. Sun, and A. Karunanidhi. 2004. Pinpointing representative portions of large Intel-Itanium; Programs with dynamic instrumentation. In *Proceedings of the 37th International Symposium on Microarchitecture*. 81–92. DOI: http://dx.doi.org/10.1109/MICRO.2004.28

J. Thomas Pawlowski. 2011. Hybrid Memory Cube (HMC). In *HOT-CHIPS*.

W. W. Peterson and D. T. Brown. 1961. Cyclic codes for error detection. *Proceedings of the IRE* 49, 1 (1961), 228–235. DOI:http://dx.doi.org/10.1109/JRPROC.1961.287814

David Roberts, Nam Sung Kim, and Trevor Mudge. 2007. On-chip cache device scaling limits and effective fault repair techniques in future nanoscale technology. In *Proceedings of the 10th Euromicro Conference on Digital System Design Architectures, Methods and Tools (DSD'07)*. IEEE Computer Society, Washington, DC, 570–578. DOI:http://dx.doi.org/10.1109/DSD.2007.83

B. Schroeder and G. A. Gibson. 2010. A large-scale study of failures in high-performance computing systems. *IEEE Transactions on Dependable and Secure Computing* 7, 4 (2010), 337–350. DOI:http://dx.doi.org/10.1109/TDSC.2009.4

Bianca Schroeder, Eduardo Pinheiro, and Wolf-Dietrich Weber. 2009. DRAM errors in the wild: A large-scale field study. *SIGMETRICS Perform. Eval. Rev.* 37, 1 (June 2009), 193–204. DOI:http://dx.doi.org/10.1145/2492101.1555372

S. Shiratake, K. Tsuchida, H. Toda, H. Kuyama, M. Wada, F. Kouno, T. Inaba, H. Akita, and K. Isobe. 1999. A pseudo multi-bank DRAM with categorized access sequence. In *Proceedings of the 1999 Symposium on VLSI Circuits*. 127–130. DOI:http://dx.doi.org/10.1109/VLSIC.1999.797260

Silicon Power 2010. *DDR3 ECC Unbuffered DIMM Spec Sheet*. Silicon Power.

Jaewoong Sim, Gabriel H. Loh, Vilas Sridharan, and Mike O'Connor. 2013. Resilient die-stacked DRAM caches. In *Proceedings of the 40th Annual International Symposium on Computer Architecture (ISCA'13)*. ACM, New York, NY, 416–427. DOI:http://dx.doi.org/10.1145/2485922.2485958

V. Sridharan and D. Liberty. 2012. A study of DRAM failures in the field. In *Proceedings of the 2012 International Conference on High Performance Computing, Networking, Storage and Analysis (SC'12)*. 1–11.

Vilas Sridharan, Jon Stearley, Nathan DeBardeleben, Sean Blanchard, and Sudhanva Gurumurthi. 2013. Feng shui of supercomputer memory: Positional effects in DRAM and SRAM faults. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC'13)*. ACM, New York, NY, Article 22, 11 pages. DOI:http://dx.doi.org/10.1145/2503210.2503257

JEDEC Standard. 2013. High Bandwidth Memory (HBM) DRAM. In *JESD235*.

Tezzaron Semiconductor. 2010. *Octopus 8-Port DRAM for Die-Stack Applications: TSC100801/2/4*. Tezzaron Semiconductor.

A. Thomasian and J. Menon. 1997. RAID5 performance with distributed sparing. *IEEE Transactions onParallel and Distributed Systems* 8, 6 (1997), 640–657. DOI:http://dx.doi.org/10.1109/71.595583

Chris Wilkerson and others. 2010. Reducing cache power with low-cost, multi-bit error-correcting codes. In *ISCA-37*.

Chris Wilkerson, Hongliang Gao, Alaa R. Alameldeen, Zeshan Chishti, Muhammad Khellah, and Shih-Lien Lu. 2008. Trading off cache capacity for reliability to enable low voltage operation. In *Proceedings of the 35th Annual International Symposium on Computer Architecture (ISCA'08)*. IEEE Computer Society, Washington, DC, 203–214. DOI:http://dx.doi.org/10.1109/ISCA.2008.22

Jei-Hwan Yoo, Chang-Hyun Kim, Kyu-Chan Lee, Kye-Hyun Kyung, Seung-Moon Yoo, Jung-Hwa Lee, et al. 1996. A 32-bank 1 Gb self-strobing synchronous DRAM with 1 GByte/s bandwidth. *IEEE Journal of Solid-State Circuits* 31, 11 (1996), 1635–1644. DOI:http://dx.doi.org/10.1109/JSSC.1996.542308

Doe Hyun Yoon, Jichuan Chang, Naveen Muralimanohar, and Parthasarathy Ranganathan. 2012. BOOM: Enabling mobile memory based low-power server DIMMs. In *Proceedings of the 39th Annual International Symposium on Computer Architecture (ISCA'12)*. IEEE Computer Society, Washington, DC, 25–36.

Doe Hyun Yoon and Mattan Erez. 2010. Virtualized and flexible ECC for main memory. In *Proceedings of the 15th Edition of ASPLOS on Architectural Support for Programming Languages and Operating Systems (ASPLOS'10)*. ACM, New York, NY, 397–408. DOI:http://dx.doi.org/10.1145/1736020.1736064

Received March 2015; revised July 2015; accepted September 2015