Configurable-ECC: Architecting a Flexible ECC Scheme to Support Different Sized Accesses in High Bandwidth Memory Systems

Hsing-Min Chen, Shin-Ying Lee, Trevor Mudge, *Life Fellow, IEEE*, Carole-Jean Wu, *Senior Member, IEEE*, and Chaitali Chakrabarti, *Fellow, IEEE*

Abstract—Designing error correction code (ECC) to guarantee strong reliability for high bandwidth memory (HBM) is imperative in high performance computers, especially for systems equipped with graphics processing units (GPUs). The design of ECC is challenging because future GPUs are expected to implement a memory subsystem supporting fine and coarse-grained data accesses to match the difference in the spatial locality of GPGPU applications. Current ECC designs, however, are developed for a fixed data fetch granularity. To have a more flexible design, we propose a novel memory protection scheme, called Config(urable)-ECC, which provides strong reliability for both fine and coarse-grained data accesses. Config-ECC consists of two tiers of ECC protection. The tier-1 code is a strong product code that can correct errors due to small granularity faults and detect errors caused by large granularity faults. The tier-2 code is an XOR-based code that is employed to correct errors incurred by large granularity faults. Config-ECC provides stronger reliability and/or lower energy consumption compared to state-of-the-art fixed 32B and 64B ECC schemes. It reduces the HBM energy by 17-21 percent while reducing the failure in time (FIT) rate by 20 times compared to a state-of-the-art fixed 64B ECC scheme with an insignificant 1.2 percent performance overhead.

Index Terms—3D DRAM, memory reliability, error control coding and GPU

1 Introduction

Graphics processing units (GPUs) play an increasingly significant role in high performance computers. Modern GPUs employed in supercomputers are equipped with very large size of memory. For example, Titan (2012) and Tianhe-2 (2013) have 109.5 and 375 TB sized memory, respectively. In order to provide greater memory density, lower access latency, higher bandwidth, and better energy efficiency to sustain the demand of modern GPUs, 3D DRAMs, e.g., high bandwidth memory (HBM) [1], are projected to be widely adopted in GPU memory systems in the near future. In fact, 3D DRAMs have already been applied in some commercial GPU products, such as Nvidia Tesla P100 [2] and AMD Radeon R9 FURY X GPU cards [3].

- *H.-M. Chen is with the Intel Corporation, Santa Clara, CA 95054-1549, and also with the School of Electrical, Computer and Energy Engineering, Arizon State University, Tempe, AZ 85287-5706. E-mail: hchen136@asu.edu.*
- *S.-Y. Lee is with Samsung, Austin, TX 78754, and also with the School of Electrical, Computer and Energy Engineering, Arizon State University, Tempe, AZ 85287-5706. E-mail: lshinyin@asu.edu.*
- *T. Mudge is with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109-2121. E-mail: tnm@umich.edu.*
- *C.-J. Wu and C. Chakrabarti are with the School of Electrical, Computer and Energy Engineering, Arizon State University, Tempe, AZ 85287-5706. E-mail: {carole-jean.wu, chaitali}@asu.edu.*

Manuscript received 20 Dec. 2017; revised 3 Sept. 2018; accepted 3 Dec. 2018. Date of publication 13 Dec. 2018; date of current version 15 Apr. 2019. (Corresponding author: Hsing-Min Chen.) Recommended for acceptance by J. D. Bruguera. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TC.2018.2886884

One of the most important issues in 3D DRAM design for GPU memory systems is the reliability aspect. While high performance GPUs are equipped with denser DRAMs, the system is likely to experience DRAM data failures more frequently. Thus, data stored in the DRAMs is typically protected by error correction codes (ECC) [4], [5], [6] to detect and recover from failures. However, promising strong reliability for 3D DRAMs is more challenging than conventional 2D DRAMs because in 3D DRAM, a data line is housed in a single bank instead of striping across multiple chips like the 2D DRAM organizations.

In addition to the essential differences between the memory architectures of 3D DRAMs and 2D DRAMs, designing an ECC scheme for GPU memory systems is more challenging than for CPUs. The reason is that the data fetch granularity from GPUs highly depends on the GPU memory hierarchy configuration. Therefore, it is difficult for HBM vendors to deliver a fixed memory product. For example, AMD GPUs, such as the RADEON series, adopt 64B as the cache line size [7], whereas Nvidia uses 128B L2 cache line size in many of their GPU products. Furthermore, many recent studies [8], [9] have shown that GPGPU applications exhibit mixed locality data access patterns over application execution. Fig. 1 shows the distribution of data fetch different sizes for GPU DRAM accesses. Each color of the bars indicates the portion of the 128B cache line before its eviction. We can observe that, within an application, the data access pattern can be mixed, demanding varying size of data. Therefore, a lot of designs have been proposed to dynamically predict and accommodate varying cache line

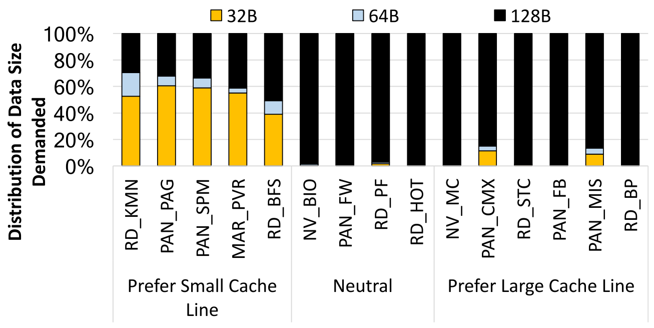


Fig. 1. The distribution of number of bytes referenced during a cache line's lifetime.

sizes for the L1 cache [9] as well as for the L2 cache [8]. Furthermore, in some GPU products, the L2 cache line size can be statically configured to operate in 2 different modes (32B or 128B) before a GPU kernel is launched [10]. This is proposed with compiler level analysis to fully exploit the varying degree of spatial locality in GPGPU applications.

Traditionally, ECC algorithms used in DRAMs were optimized for fixed sized data accesses. Examples include CRC-16 for 32B accesses in [1], symbol-based code for 64B accesses in [11], [12]. However, applying an ECC scheme that is optimized for a single data access size leads to suboptimal memory reliability or higher energy consumption for the other data access sizes. For instance, an ECC algorithm optimized for 64B access leads to additional energy consumption if the GPU data fetch size is 32B. Alternatively, an ECC algorithm optimized for 32B data results in lower reliability compared to one that results in been designed for 64B data. Clearly, to better support the diversity of cache configurations in GPUs and the performance feature of the variable data fetch granularity runtime support for future GPUs, there is a strong need to develop an adaptive and flexible ECC scheme.

In this paper, we propose Config(urable)-ECC, an ECC scheme that not only supports a diverse set of cache hierarchy configurations in different GPU cards in the static mode but is also designed to handle variable data fetch granularities at runtime in the dynamic mode with strong reliability and/or low energy consumption. Config-ECC is based on a two-tiered ECC code: The tier-1 code is a product code [4] that has strong error detection and correction capabilities; the most important selection of product code is that it allows the different access sizes. It can correct errors due to small granularity faults (single bit, column, and TSV failures) and detect errors with large granularity fault (row or bank failures) with very low silent data corruption rate. The tier-2 code, which is based on an XOR structure and stored in the data banks, is launched for correcting errors due to large granularity faults.

Config-ECC is built around a core 32B ECC scheme that can be easily extended to support 64B and 128B accesses. The product code structure of Config-ECC helps address the mismatch between data access size and fixed size ECC scheme. Basically, the inner code is optimized to have very strong detection capability for 32B access and the combination of inner code and outer code provides for very strong reliability for 64B access. Config-ECC improves the silent data corruption rate (SDC) by 200 times compared to the 32B ECC scheme in the HBM standard and by 20 times compared to the latest 64B ECC schemes with a negligible 1.2 percent performance overhead [11], [13]. For applications that prefer small cache line size (32B), Config-ECC has significant energy reduction (21 percent) compared to the 64B and 128B ECC schemes and also provides stronger reliability than that of a fixed 32B ECC scheme.

Our main contributions are as follows

- We designed a two-tiered ECC scheme to provide strong error protection for mixed 32B, 64B and 128B accesses in 3D HBM memory systems used in high performance GPUs. These three ECC schemes share the same core structure and can be configured statically and dynamically to support different sized accesses.
- Compared to a fixed 32B ECC scheme, Config-ECC significantly increases the reliability by 200 times for both static and dynamic modes with only an insignificant 1.2 percent performance overhead.
- Compared to a fixed 64B ECC scheme, Config-ECC increases the reliability by 20 times when the access size is 64B or larger. Also, Config-ECC can choose to only read 32B of data to reduce the energy by 21 percent in the static mode and 17 percent in the dynamic mode.

The rest of the paper is organized as follows: Section 2 describes the basics of a 3D DRAM memory system followed by error characteristics and motivation for a flexible ECC design for GPU system. Section 3 gives the high level overview of our proposed Config-ECC and Section 4 presents the detailed design for 32B accesses. The 32B access scheme is extended to support the static and dynamic mode accesses in Section 5. The reliability and simulation infrastructure are given in Section 6 and the results are shown in Section 7. Finally, Section 8 concludes this paper.

2 Background and Motivation

2.1 3D High Bandwidth Memory Architecture

In this work, we focus our ECC design for HBM[1] since HBM has been integrated with commercial GPU products. Each layer in HBM houses multiple DRAM banks, each of which consists of two sub-banks. A group of banks share a channel. For instance, in the latest HBM standard [1], 8 or 16 banks share a single channel and a single layer consists of two separate channels. Within a DRAM bank, DRAM cells are organized into rows and columns, similar to a conventional 2D DRAM bank. A data line is stored in the same bank in HBM in contrast to a data line being striped across multiple banks in conventional 2D DRAMs.

The HBM architecture is shown in Fig. 2. For each read or write operation, data is accessed from a single bank. The basic access unit is 32B in HBM1 [14] while the access unit can be 32B or 64B in HBM2 [1]. The I/O for each channel is 128b with additional 16b for ECC in [1], [15]. While HBM2 supports two modes: the legacy mode and the pseudo channel mode, here we focus on the pseudo channel mode which

1. Since HBM2 is the latest HBM standard, we use HBM and HBM2 interchangeably throughout this paper.

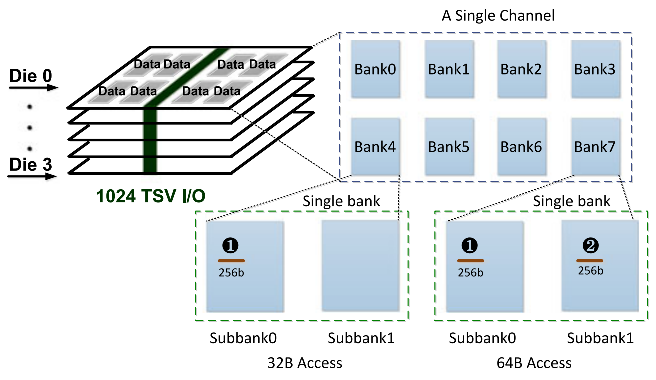


Fig. 2. 32B versus 64B accesses in HBM (Pseudo channel mode).

has lower actiaion power and higher bandwidth. In this mode, a single channel is divided into two sub-channels each of size 64 bits and so a 32B access is transferred through 4 bursts (2 cycles). In a 64B access, two sub-channels are activated at the same time and the 64B access is also transferred through 4 bursts (2 cycles).

The ECC bits are stored in the data banks instead of being housed in another die as in [13]. In each access, the ECC bits are read along with the data bits. We assume that the storage overhead of ECC is 12.5 percent based on the HBM standard outlined in [1] and [15].

2.2 3D DRAM Errors

DRAM errors can be broadly classified into soft errors and hard errors [16], [17]. Soft errors are caused by transient faults which occur randomly and cause incorrect data to be read from a memory location; they disappear when the error location is overwritten. Hard errors are caused by permanent faults or intermittent faults. A permanent fault causes a memory location to consistently return an incorrect value. Note that a single fault can result in multiple error instances [16].

DRAM errors can also be classified into those that are due to small granularity faults such as single bit or single column that account for 62.8-84.8 percent of all faults, and large granularity faults such as row and bank failures [16], [18], [19]. 3D DRAM also has errors due to TSV failures [12], [13], which fall under small granularity faults. A single row or bank failure leads to multiple errors which can be seen as burst errors in a data line. To minimize the miss-error detection of errors due to large granularity faults, the ECC should also have strong burst error detection capability. In this paper, we analyze the capability of the memory system to handle errors due to small granularity and large granularity faults (transient and permanent).

Real-world field data from [16] provided DRAM failure rates as failures in time (FIT) in a single 1 Gb DRAM chip/device. In this work, we assume that each die consists of two channels and each channel supports 16 banks with a capacity of 4 Gb (8 Gb per die). We use the FIT rate from [16] and scale the failure rate proportionally to the size of DRAM chips. For each ECC scheme, we report the final detected and corrected error rate (DCE), detected but uncorrected error rate (DUE) and silent data corruption rate (SDC) [20] of the decoded results; the corresponding analysis is given in Section 7.

2.3 Related Work

Several ECC schemes have been proposed for 3D DRAM systems. We discuss a few of them here.

RATT-ECC [11] uses RS code as the tier-1 code to correct all errors due to small granularity faults and has very strong detection capability for errors due to large granularity faults. It relies on a simple tier-2 code to recover from errors due to large granularity faults. If a spare bank is needed, the RS code is shortened to support the spare bank.

E-RAS [12] proposed a two tier error correction scheme with higher than 12.5 percent storage overhead. The first tier uses a symbol-based ECC code to correct errors due to small granularity faults, while the second tier is an XOR-based correction code (XCC) to correct the detectable but uncorrectable errors due to large granularity faults. It proposed a strategy to deal with permanent TSV (or row) failures by using spare TSVs (or rows). However, it does not have very strong detection capability for errors due to large granularity faults.

Citadel [13] also uses two tier ECC protection to handle errors due to small and large granularity faults. It uses CRC-32 to provide strong error detection capability. After errors are detected, it relies on multiple levels of parity (3DP) to recover errors. It provides for two spare banks to handle permanent bank failures. While Citadel is optimized to handle errors due to permanent faults (TSV, row, bank), it has a very large overhead for correcting errors due to transient faults including those that affect only a single bit.

Parity-Helix [21] focuses on the tier-2 code design compared to [11], [12], [13], which focus on tier-1 code. Specifically, it protects against whole die failure or a single channel failure. The design is based on the RAID-5 algorithm in a helical fashion with the parity bits being generated on data gathered from banks in a die and across dies.

For 2D DRAM ECC schemes, here, we summarize a few that can support multiple access sizes or reconfigurable structures. For instance, VL-ECC [22] reconfigures the data protection size so that the most significant bits get higher protection compared to the other bits. Since the lower significant bits can still be erroneous, such a scheme is not acceptable for high performance computing applications. CLEAN-ECC[23] and DGMS [24] support access sizes ranging from 16B to 64B. In [23], the ECC is optimized for correction of a 32b block since a single device failure leads up to 32b errors in a data line in 2D DRAM. DGMS [24] uses multiple SEC-DED codes to protect a single data line, specifically, they use 8b to protect every group of 64b of data. Both CLEAN-ECC [23] and DGMS [24] are not strong enough to handle 3D DRAM errors.

2.4 Motivation for an ECC Design Supporting GPU's Dynamic Varying Data Fetch Sizes

General-purpose GPU kernels exhibit dynamically-varying data access patterns and different degrees of spatial locality. Thus, a static, general memory configuration cannot be optimal across all kinds of GPGPU applications. Consequently, the cache line size in today's commercial GPU products has a diversity of configurations. For example, the cache line size of NVIDIA Fermi GPUs [25] is 128B, the cache line size of Intel Gen9 GPUs and AMD Graphics [7], [26] is 64B, and the data fetch granularity of NVIDIA Maxwell GPUs [27] is

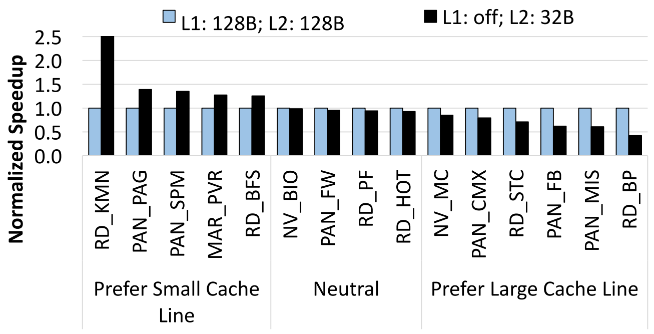


Fig. 3. The normalized execution time speedup with different two cache configurations: 128B cache line size for both L1 and L2 caches versus L1 cache disabled and 32B cache line size for the L2 cache.

32B with two different L2 cache line sizes of 32B and 128B. The device driver of NVIDIA Tesla GPUs can be modified to launch kernels with two cache configurations. When the L1 cache is turned on, the L1 as well as for the L2 cache lines are 128B. When the L1 cache is turned off, the data fetch size of the L2 cache becomes finer-grain 32B.

Fig. 3 shows the performance characterization results for GPGPU applications running with the different cache configurations on a Nvidia Kepler-based GPU. The blue bars represent the performance of our baseline system with the L1 cache enabled, as such, the L2 cache operates at the 128B data fetch granularity whereas the black bars represent the performance speedup when the L1 cache is disabled, resulting a finer-grained 32B data fetch size. Applications in the first category receiving performance speedup by an average of 1.6X prefer a finer data fetch granularity while other applications prefer a coarse-grained data fetch size because of the relative well-exploited spatial locality. While applications with high spatial locality favor the 128B data fetch size, applications with poor degree of spatial locality prefer 32B access granularity.

Also, within a GPU kernel, optimal data fetch granularities can vary as well. Arunkumar et al. demonstrated a strong correlation between an issuing instruction, its degree of memory divergence and the optimal data fetch size [9] whereas Rhu et al. showed a dynamic data fetch granularity design can significantly improve the performance and energy efficiency of GPUs [8]. Our own characterization study (Fig. 1) illustrates a similar trend for a large set of GPGPU applications. Since the optimal data access granularity changes during the kernel execution at runtime, an optimized GPU design needs to be able to predict and apply the optimal memory configuration to improve resource utilization and the performance of the GPGPU application.

Traditional DRAM ECC algorithms are developed and optimized for fixed sized data accesses. Applying an ECC algorithm that is optimized for a certain data fetch size causes configuration mismatch, that leads to sub-optimal reliability and energy consumption optimization. For instance, an ECC algorithm optimized for 128B data accesses results in data over-fetch, if the GPU cache line size is 32B, leading to additional energy consumption. This is because to perform error correction for a single 32B request, the entire 128B of data needs to be fetched. On the other hand, with an ECC algorithm that is optimized for 32B data, when data requests from the GPU are of 128B, the degree of reliability guarantee offered by the 32B-optimized ECC algorithm is lower than that of an ECC algorithm that is tailored made for 128B accesses. We refer to mismatches between GPU data fetch sizes and ECC algorithms as *configuration mismatches*.

To address the issues due to configuration mismatch, we propose Config-ECC, a mechanism to provide a strong reliability guarantee while minimizing data fetch energy consumption for the GPU memory subsystem for a varying granularity of data accesses. The proposed Config-ECC supports two operation modes:

- *Static Operation Mode* which targets at tackling the mismatch problem by setting to protect small or large data granularity depending upon the need at an application level.
- *Dynamic Operation Mode* which aims to resolve the mismatch issue by interleaving small and large sized ECC protection for supporting the mixed locality behavior during runtime.

3 Overview

In order to handle both small and large granularity faults in HBM systems that support different sized data accesses (32B, 64B and 128B), we propose, Config-ECC, a flexible ECC architecture. Config-ECC is based on a two-tiered ECC scheme, where the tier-1 code is designed to correct errors due to small granularity faults and to detect errors due to large granularity faults and the tier-2 code is designed to perform error correction only when tier-1 code detects errors due to large granularity faults.

This scheme is very different from existing two-tiered 3D DRAM ECC schemes [11], [12], [13], which all optimize for 64B access. When GPU requests different sized data, these schemes have high energy overheads, as will be demonstrated in Section 7. The existing ECC schemes proposed in 2D DRAM [28], [29], [30] do not have the capability to correct errors due to large granularity faults in HBM systems and cannot be used. Virtualized-ECC [28], which provides single symbol correction and double symbol detection, is not strong enough for errors due to large granularity faults.

To support strong protection and different sized accesses, we design a core ECC structure based on 32B accesses and extend it to support 64B and 128B accesses. This is achieved by using an inner code and an outer code analogous to product codes [4]. The overview of Config-ECC is shown in Fig. 4. In the dynamic mode, for 32B accesses, we only use the inner code to protect 32B of data; the outer code is launched when the inner code detects errors. For 64B accesses, both inner code and outer code are used, resulting in better reliability compared to the 32B accesses. For 128B accesses, two separate 64B decoding units are used resulting in the same protection capability as 64B accesses. In the static mode, we add interleavers to increase the burst error detection capability. Recall that large granularity faults manifest as burst errors and adding interleavers is a low cost method of improving burst error detection capability. For 64B accesses, an interleaving unit is used to spread the erroneous bits in two 256b of data to two inner code decoding units. Similarly, for 128B accesses, the interleaving unit is used to spread the erroneous bits of four 256b of data to four inner code decoding units.

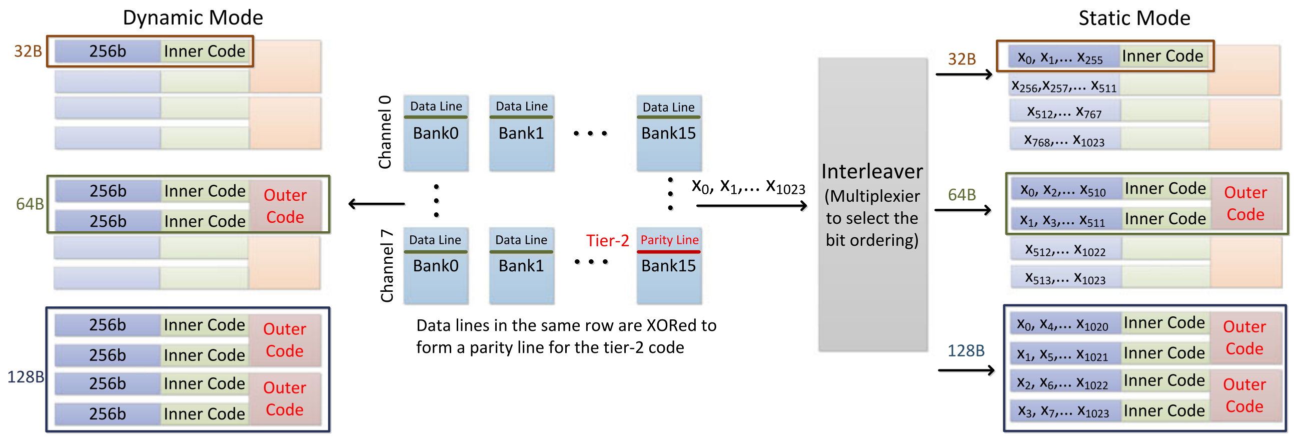


Fig. 4. Overview of config ECC.

3.1 Design of Tier-1 Code

In our system, the tier-1 code has the following requirements. First, it should have strong detection capability for the different sized accesses. Specifically, it should detect all errors due to small granularity faults and have very low SDC rates for errors due to large granularity faults. This feature is very important since the tier-2 code is only activated when the tier-1 code detects errors.

Second, the tier-1 code should correct all errors due to small granularity faults. Recent work on DRAM reliability indicates that errors due to small granularity faults account for more than 70 percent of the total faults [17], [18]. Since the tier-2 code has larger latency and consumes more energy to correct errors, errors due to small granularity faults should be corrected by the tier-1 code. Third, the storage overhead of the tier-1 code should be constrained to 12.5 percent, as indicated in the HBM standard [1].

Since the ECC storage overhead is 12.5 percent for the tier-1 code design, each 32B (=256 bits) of data is protected by 32 ECC bits. These 32 ECC bits are stored along with the data in the same bank. Of these 32 bits, N parity bits are reserved for the inner code and the remaining $(32-N)$ bits are reserved for the outer code. The choice of whether more bits should be allotted to inner code or outer code is dictated by the following consideration. If we put more bits in the inner code, it would have strong detection capability but the outer code might not be able to support correction. On the other hand, if we put more bits in the outer code, the inner code might have weak detection capability for errors due to large granularity faults and might lead to miss-error events. So we choose to allot more ECC bits to the inner code to support strong detection capability.

Inner Code. We allocate N parity bits to support correction or detection of 32B data. If the inner code is designed to provide for both correction and detection, its overall detection strength would be lower [4]. So we choose to use the N parity bits for only detection, so that the tier-1 inner code has strong detection capabilities. Designing the inner code for strong detection guarantees that errors due to small and large granularity faults be detected, producing very low SDC rates. If the inner code detects errors, the tier-1 outer code can be used to correct errors due to small granularity faults and the tier-2 code can be used to recover errors due to large granularity faults.

Outer Code. Of the 32 ECC bits, if N bits are allotted for tier-1 inner code, we only have $32-N$ bits for the outer code. If N is (say) 24, there are not enough bits to design an outer code with the desired detection and correction capabilities. So we combine ECC bits associated with two sets of 32B data. Specifically, we combine $2\times(32-N)$ ECC bits to form the parity bits of the outer code. Such a scheme necessitates tier-1 code to be split across two 32B access units. The performance penality due to this splitting is very small, as will be described in Section 7.

The outer code has to correct all errors due to small granularity faults such as single bit, single column and single TSV faults. This is achieved by a symbol based code, as described in Section 4.1.

3.2 Tier-2 Code

Once the errors in the data line are flagged as errors due to large granularity faults (that cannot be corrected by the tier-1 code), the tier-2 code is invoked. The tier-2 code corrects errors due to large granularity faults caused by a single row/bank failure in a die-stacked DRAM. Our tier-2 code is based on an XOR-correction code as in [11], [12], [13], [21]. According to the failure characteristics described in Section 2, multiple bank failures in the same die are less likely to occur and we focus on correcting errors due to one bank failure. In our 3D DRAM structure, there are 128 banks. Of these, one bank is reserved for storing the parity of data in the remaining 127 banks. Thus when tier-2 code is launched, 126 data lines and 1 parity line (in the same row location) are read out and XORed to generate the correct data line. Our tier-2 code has a storage overhead of 1/127. Since the tier-2 code parity line is also housed in the data banks, the real data size in a HBM stack is reduced. Note that we could have chosen to XOR 64 banks or 32 banks for lower decoding latency (read 64 or 32 lines per correction) but at the expense of lower storage capacity.

4 Details of 32B Config-ECC

This section describes the detailed design of our ECC scheme for fixed 32B data fetch size HBMs. We first describe the design of the inner and outer code in Section 4.1. Next, we summarize the error detection and correction operations and the decoding flowcharts for 32B accesses in Section 4.2.

TABLE 1
Inner Code and Outer Code Combination Candidates

Code	N (inner)	2x(32-N) (outer)	ECC codes (inner + outer)
1	16	32	2 CRC-16 + RS (73,69)
2	20	24	2 CRC-20 + RS (72,69)
3	24	16	2 CRC-24 + RS (72,70)
4	26	12	2 CRC-26 + SEC-DED (575,564)
5	28	8	2 CRC-28 + N/A

4.1 Design of the Tier-1 Code

As described earlier, the tier-1 code consists of an inner code which should have strong error detection capability and an outer code which should be able to correct single symbol errors. For the tier-1 inner code, both Reed-Solomon (RS) [31] symbol based code and cyclic redundancy code (CRC) [32] are good candidates since they have strong detection capabilities. Low density parity check (LDPC) [4] is not an appropriate candidate here although it has been successfully used in SSD/disk. The reason is that LDPC is designed for large block size (2 to 8 KB in SSD) while DRAM access size is 256b or 512b. It also has large decoding latency (10 - 20 cycles), which is unacceptable for DRAM systems and its good performance comes from soft decision decoding [4], which cannot be supported in DRAM systems.

Recall that we allocate N parity bits for the inner code, where N varies from 0 to 32, and 2x(32-N) bits for the outer code. A large value of N implies that the inner code will have strong detection capability but the outer code could end up with poor detection/correction capability. On the other hand, a small value of N implies that the inner code will have weak detection capabilty and the outer code will have strong detection/correction capability. Thus the reliability performance of the tier-1 code depends on the choice of N.

In order to select the value of N that is best suited for our ECC scheme, we analyze the performance of 5 candidate schemes listed in Table 1. If N = 16 (Candidate 1), the SDC rate is 2^{-16} (for 32B access) which is high. The outer code now has 32b and can support double symbol correction. However, double column or double TSV failures happen very infrequently and so allocating 32 bits to the outer code is an overkill. Candidate 2, corresponding to N = 20, has SDC rate of 2^{-20}, which is lower than Candidate 1, as expected. The outer code now provides single symbol correction and double symbol detection. Candidate 3, corresponding to N = 24, has SDC rate 2^{-24} (for 32B access) and the corresponding outer code can support single symbol correction. This is strong enough to handle errors due to single TSV failure or single column failure. Candidate 4, corresponding to N = 26, has better SDC rate compared to the N = 24 case but the outer code now only has 12 bits. The only available code is SEC-DED (575,564), which cannot correct errors due to a single TSV failure and is thus not appropriate. Candidate 5, corresponding to N = 28, only has 8b for the outer code and thus is not strong enough for even a single bit correction. From this analysis, we conclude that the outer code needs a minimum of 16b (2×8 symbols to perform single symbol correction), which means N should be smaller than or equal to 24. So we choose N = 24 for Config-ECC. Its inner code has strong detection capability and its outer code can support single symbol correction. Together, the tier-1 code can correct all small granularity faults and provide good detection capability for large granularity faults.

For the outer code, we choose RS (72,70) over finite field GF(2^8). It can provide single error correction because its minimum distance is 3 [4]. Since the errors due to small granularity faults might occur in the data bits as well as the CRC parity bits, the RS (72,70) code is used to protect both data bits and CRC parity bits.

For the design of the inner code with N = 24, we choose CRC-24 over RS (35,32) code in GF(2^8). While both codes use 24b ECC to protect 256b data, RS (35,32) can detect three symbol errors among 35 symbols (8 bits per symbol) but cannot guarantee detection of 24 consecutive bit errors in a 280 bit (35×8) codeword. In contrast, CRC-24 can detect any 24 consecutive bit errors in the 280 bit codeword and since it has minimum distance of 6 [33], it can detect up to 5 random bit errors. Overall, CRC-24 has stronger detection capability in terms of random bit error detection and burst error detection compared to RS (35,32). Furthermore, implementation of CRC is straight forward and so for the tier-1 inner code, we choose the CRC-24 code. Specifically, we choose the CRC polynomial 0xBD80DE from [33], [34]. This polynomial has the largest minimum distance of 6 among all CRC-24 codes and also detects all odd number of errors.

4.2 Error Correction & Detection

The flowchart for 32B access and the corresponding ECC decoding steps are summarized in Fig. 5. The decoding process is as follows: (i) if the CRC-24 unit reports error-free, the RS decoder is not launched, (ii) if CRC unit reports errors (the first pass), an additional read is issued to retrieve the second 256b data. If only the first CRC unit reports errors, the RS decoder is launched to perform single symbol correction. After the correction, the two CRC units check whether there are any remaining errors. If two CRC units report errors in the beginning, RS decoder is not launched and tier-2 code is activated. Next we analyze the error correction and detection capability of the 32B access scheme.

Errors Due to Small Granularity Faults. CRC-24 will successfully detect errors due to a single bit failure and a single column failure. A single TSV failure results in 4-bit error, which can be detected by one CRC-24 since CRC-24 has minimum distance of 6. A small granularity fault only leads to one symbol error in a RS codeword, which can be corrected.

Errors Due to Large Granularity Faults. After the RS (72,70) decoder performs correction, two CRC-24 codes recheck the CRC bits. If any one of the two CRC-24 codes flags errors, the tier-2 code has to be activated to correct errors due to large granularity faults.

Silent Data Corruption. There are two situations when the tier-1 code suffers from silent data corruption. First, when the CRC decoder detects errors, the RS code performs error correction and the two CRC codes miss-detect the corrected words. If the RS code performs correction, the two CRC codes recheck the decoded data again. If two CRC codes miss-detect the errors, an SDC event occurs.

Second, the CRC miss-detects errors at the first detection; RS(72,70) will not be launched and the RS decoder cannot

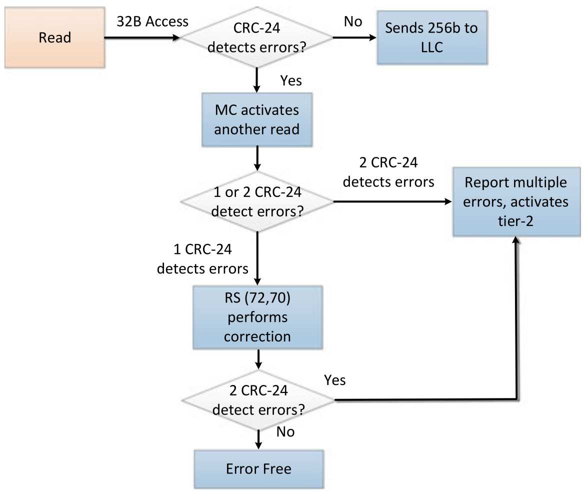


Fig. 5. 32B access flowchart.

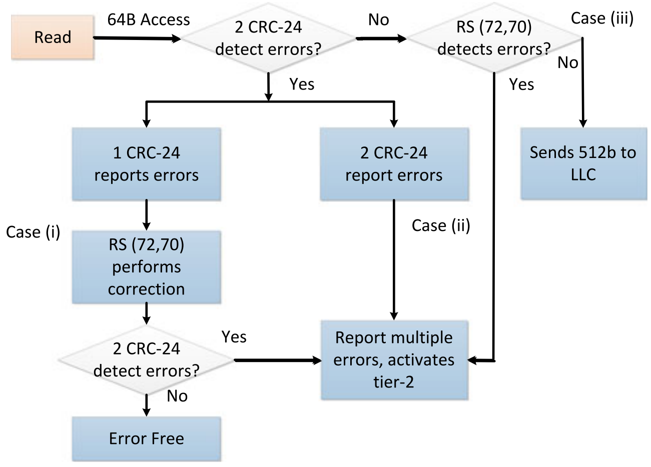


Fig. 6. Decoding flowchart for 64B access.

help in this case. Hence, it is very important that tier-1 inner code to have very strong detection capabilities.

5 Extensions of Configurable-ECC

In this section, we explain how our 32B ECC scheme can be easily extended to support Dynamic Operation Mode (Section 5.1) and the Static Operation Mode (Section 5.2).

5.1 Dynamic Mode Protection

5.1.1 64B Access Size

In this case, two 32B of data and two 32b of ECC are read out from two sub-banks and sent to the tier-1 decoder. The tier-1 decoder checks for errors using CRC-24 decoder, corrects single symbol error using RS (72,70) and launches tier-2 if errors are detected.

The decoding steps for 64B access are as follows: (i) if only one CRC-24 code reports errors, the RS (72,70) decoder performs single error correction, followed by CRC-24 detection as in the 32B access scheme, (iii) if two CRC codes report errors, RS decoder is not activated and the tier-2 decoder is launched. (iii) if two CRC-24 codes report error-free, the RS (72,70) decoder performs double error detection.

If two CRC codes report errors at the same time, it means that the errors must not have been caused by the small granularity faults and thus tier-2 code is triggered. The decoding case (iii) is different from the 32B access scheme where the RS (72,70) decoder was used for single symbol correction. Since RS (72,70) code can perform either single error correction or double error detection, once two CRC codes report error-free, the RS decoder activates double error detection to avoid possible miss-error detection by two CRC codes. The decoding flow chart is presented in Fig. 6. Our analysis of the error performance of the 64B access scheme is as follows.

No Errors. When there are no errors, 2 CRC-24 codes and RS (72,70) report error free and the memory controller sends the 512b to the lower level cache.

Errors Due to Small Granularity Faults. A single bit failure and a single column failure lead to one bit error which can be fully detected by one CRC-24 code. A single TSV failure results in 4 bit errors in one CRC-24 code and since CRC-24 has minimum distance of 6, the CRC-24 code can fully detect this error event. Thus, when there is a small granularity fault, only one CRC-24 code report errors. The small granularity fault only causes one symbol error in an RS codeword, and can be fully corrected.

Errors Due to Large Granularity Faults. When two CRC-24 codes declare errors, tier-2 code is launched. This is because the number of errors is beyond the correction capability of the RS code.

Silent Data Corruption. Silent data corruption can occur in the following scenarios. First, two CRC-24 codes declare error free, and RS (72,70) also declares error free. Second, one CRC-24 code declares errors, RS (72,70) performs correction and two CRC-24 codes declare error free.

5.1.2 128B Access Size

Here, 128B of data is split into two 64B of data and fed to two Config-ECC 64B units and thus the decoding flow is the same as 64B access (see Fig. 6). Consequently, the 64B access scheme and the 128B access scheme have identical reliability.

5.2 Static Mode Protection

Errors due to large granularity faults manifest as burst errors. So we propose to add interleaving units to improve the overall error detection capability.

5.2.1 64B Access Size

We interleave 32B data from each sub-bank and then encode it with CRC-24 code. Such a scheme increases the burst error detection capability without the use of a larger CRC code which has higher hardware complexity. The interleaved CRC decoding procedure is based on the method in [35]. The bit sequence of two 32B data lines are redistributed as follows. In each data line, if the bit position modulo 2 is 0, it is sent to the first CRC-24 unit; if the bit position modulo 2 is 1, it is sent to the second CRC-24 unit. After interleaving, the decoding steps are the same as dynamic Config-ECC 64B scheme.

5.2.2 128B Access Size

128B of data are redistributed before sending to four CRC-24 units. Assume that for the first 64B access, 32B are read

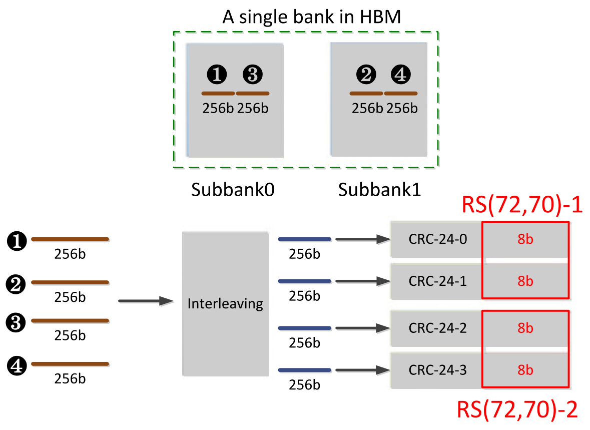


Fig. 7. Configurable-ECC for 128B access: Static mode.

from sub-banks 0 and 1 as shown in Fig. 7. If the bit position modulo 4 is 0, 1, 2 and 3, then that bit is send to CRC-24 unit 0, 1, 2 and 3, respectively. Now for the second 64B access, the mapping function is a little different. For data lines 3 and 4, if the bit position modulo 4 is 1, 0, 3 and 2, these bits are sent to CRC-24 units 0, 1, 2, and 3, respectively. This mapping function was designed to avoid 8 bit errors due to a single TSV failure be sent to the same CRC unit.

The decoding steps for static 128B access are as follows: (i) if one or two CRC-24 units report errors, the RS decoder performs single error correction, (ii) if more than two CRC-24 decoder report errors, RS decoder is not activated and tier-2 decoder is launched, (iii) if four CRC units report error free, the RS decoder performs double error detection. The decoding flowchart is given in Fig. 8. Small granularity faults will fall in case (i). The large granularity faults fall in either case (ii) or case (iii).

6 Methodology

We introduce the experimental setup used to evaluate the reliability, performance and energy consumption of Configurable-ECC in this section.

6.1 Setup for Reliability Evaluation

To measure system-level failure probability, we use Monte Carlo based simulations. The number of trials is at least 1 billion. Since the number of simulations is very large, we

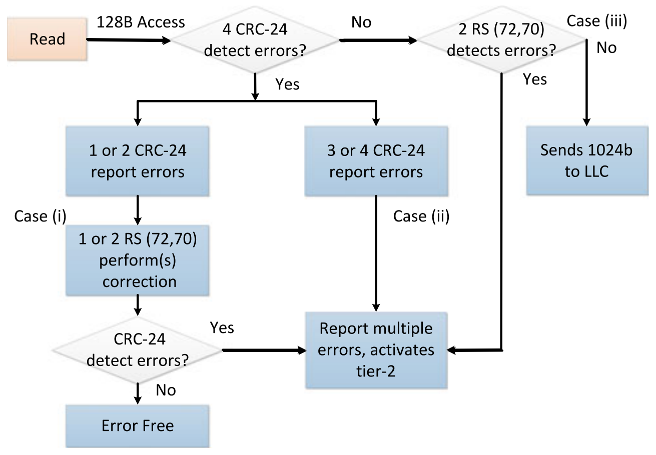

Read
128B Access
4 CRC-24 detect errors?
No
2 RS (72,70) detects errors?
Case (iii)
No
Yes
Yes
1 or 2 CRC-24 report errors
3 or 4 CRC-24 report errors
Sends 1024b to LLC
Case (i)
Case (ii)
1 or 2 RS (72,70) perform(s) correction
Yes
CRC-24 detect errors?
Report multiple errors, activates tier-2
No
Error Free

Fig. 8. Decoding flowchart for 128B access: Static mode.

TABLE 2
GPU and HBM System Configuration

GPU Architecture	
# of SMs,	15
Max. # of Wraps per SM	48
Max. # of Blocks per SM	8
# of Warps per SM	32
# of Schedulers per SM	2
# of Registers per SM	32768
L1 inst. cache	2 kB per SM, 128B line size 4 sets, 4-ways
L2 cache	768 kB, 128B line size 32 sets, 8-ways, 6 banks
Scheduler	Greedy-then-oldest (GTO)
HBM Configurations	
Capacity	4GB per stack
# of Channels	8
# of banks	16 per channel
I/O	144 per channel
Memory clock	0.5 GHz
Row Size (pseudo channel mode)	1 kB for 32B access 2 kB for 64B access
HBM Memory Timing (cycles)	tRC=24, tRCD=7, tRP=7, tCL=7, tRAS=17, tRRD=3, tCCD=2, tWL=1, tRTP=2

use the Mersenne Twister algorithm as the pseudo number generator to generate error patterns that ensure randomness [36]. We inject faults into the 3D HBM DRAM system and analyze the results after ECC decoding. Our procedure to calculate the reliability of each ECC scheme is similar to others [11], [12], [13], [21].

For errors due to single bit and single column failure, we assume that there is one single bit error per access (32B, 64B or 128B). We assume that the number of bit errors due to a single TSV failure is 4, 8 and 8 for 32B, 64B and 128B access, respectively. Prior work [17] indicated that a row or bank failure could result in 3 to 31 random bit errors. While [12] simulated this error event, [21], [37] pessimistically assumed that half of the bits in each data line are erroneous. In our simulations, we inject 3 to 128 random bit errors in a data line to model errors due to large granularity faults.

We ECC decoding results to check whether the errors are detectable and correctable errors (DCE), detectable but uncorrectable errors (DUE) or result in silent data corruption (SDC) [20]. Finally, we also use the raw FIT rate provided from [11] and give the final FIT rates of our three Configurable-ECC schemes in Table 7.

6.2 System Simulation Infrastructure

To evaluate the performance of our ECC schemes in terms of performance and energy consumption, we use GPGPU-Sim (version 3.2.2), a cycle-level performance simulator of a general purpose GPU architecture. The key micro-architectural parameters of the baseline configuration are summarized in Table 2. The GPGPU-Sim simulator is also used to generate DRAM traces for energy calculations. To generate the DRAM traces, we assume that the L2 cache is equipped with a perfect spatial locality predictor. The assumption of perfect spatial locality predictor was done to simplify the model. Such an assumption results

TABLE 3
Benchmarks for Timing and Energy Evaluation

Abbr	Application	Dataset
RD_KMN	K-Means [41]	494020 objects
PAN_PAG	Page Rank [40]	1M data entries
PAN_SPM	Sparse Vector Multiplication [40]	1M points
MAR_PVR	Page View [39]	1M pages
RD_BFS	Breadth-First-Search [41]	65536 nodes
NV_BIO	Binomial Options [38]	512 Options
PAN_FW	Floyd Warshall [40]	256(V), 16K (E)
RD_PF	Particle Filter [41]	28x128x10 nodes
RD_HOT	Hotspot [41]	512x512 nodes
NV_MC	Monte Carlo [38]	256 options
PAN_CMX	Graph Coloring Max [40]	ecology
RD_STC	Streamcluster [41]	32x4096 nodes
PAN_FB	Floyd Warshall Block [40]	256(V), 16K (E)
PAN_MIS	Maximal Independent Set [40]	ecology
RD_BP	Back Propagation [41]	65536 nodes

in fewer number of data bits being read from DRAM, thereby saving energy. In other words, this assumption show cases the best energy saving provided by our scheme. For each DRAM trace, we record the issued time of DRAM requests and number of bytes that have been referenced during a cache line's lifetime at the L2 cache controller. We use the DRAM traces to compute the energy consumption of the different ECC schemes. We select a wide range of benchmarks from Nvidia SDK [38], Mars [39], Pannotia [40], and Rodinia [41], [42] benchmark suites to represent the diversity of memory access patterns in GPUs. The details of benchmarks are listed in Table 3.

6.3 Config-ECC Decoding Latency

The decoding latency of Config-ECC is due to CRC-24 (for 32B access) and CRC-24 + RS (66,64) (for 64B or 128B access). CRC-24 can be implemented as a lookup-table and so its latency is very low. Since HBM2 operates at 500 MHz, one cycle period is 2 ns. The table-lookup method can be optimized for one cycle delay decoding. Hence, if the read request is 32B, the memory controller needs one additional cycle to generate the redundancy bits. When the access is 64B, two CRC-24 codes and one RS code will be launched at the same time and we assume these two units can be decoded parallelly. The decoding latency of RS (66,64) is also quite small. Its syndrome calculation unit (critical path) is only 0.42 ns using 28 nm library. RS code doesn't need to wait for the decoding results of CRC-24 because it just performs detection. The memory controller only has to check whether the two syndrome vectors are zero vector or not. Thus the ECC decoding overhead is only one cycle for 32B, 64B and 128B accesses; we add this one additional cycle in our timing performance simulation.

Use of tier-2 code to correct large granularity faults comes with a large overhead. This is the case for the other existing schemes [11], [13] as well. Specifically, tier-2 code of Config-ECC is an XOR-based code which requires 127 additional reads and thus has large decoding latency and large energy overhead. We find that if 127 reads are distributed across 8 channels and each read is from a different bank in a channel, the latency is around 400 ns. Since the large granularity faults are very rare, the latency of tier-2 code can be well hidden.

6.4 HBM Energy Modeling

We model HBM energy consumption using DRAMSim2 [43], a cycle-level accurate memory system simulator. For activation/precharge energy, we use the latest value provided by CACTI-3D [44] and plug this value into the simulator. For 1 kB page size, we assume that the row energy is 1.5 nJ [21] and for 2 kB page size, we scale the row energy to 3 nJ. Similarly, we assume 32B read/write energy to be 4 nJ based on CACTI-3D. Since there is no publicly available data for HBM background and refresh current, we only show the results when only activation/precharge and read/write energy are considered.

6.5 Summary of Read/Write Accesses of the Dfiferent Schemes

For better understanding of the performance and energy results, we summarize the different types of accesses for 32B ECC [1], 64B ECC [13], 128B ECC, and Config-ECC. The 32B ECC scheme is based on CRC-16 [1], the existing 64B ECC scheme is Citadel [13],[2] and the 128B ECC scheme was designed exclusively for 128B data. Such a 128B scheme would have very high reliability for the same 12.5 percent parity storage.

32B Read Request. Config-ECC and [1] have no additional reads but [11], [13] have an additional 32B read since their ECC are based on the design of 64B access. Any 128B ECC scheme would have to read additional 96B of data to perform ECC decoding.

32B Write Request. There is no overhead for [1]. However, Config-ECC has to read an additional 32B of data (if it is not cached) while [11], [13] have to peform a 64B read first and then write 64B back. The 128B ECC scheme needs to perform a 128B read and write 128B data back.

64B Read Request. There is no overhead for Config-ECC, [1] and [11], [13]. The 128B ECC scheme needs to read another 64B of data.

64B Write Request. There is no overhead for Config-ECC, [1] and [11], [13] as well. The 128B ECC scheme needs to read an additional 64B of data and write 128B data back.

128B Request. For both read and write requests, there is no addtional overhead for any of the ECC schemes.

Since all ECC schemes are two-tiered ECC schemes, we assume that the latency of the updates for tier-2 code is well hidden by the ECC cache as in [11], [12], [13]. The updates of tier-2 code degrade the timing performance by 3 percent and we include this in our simulation results.

7 Results and Analysis

7.1 Timing Performance Analysis

To evaluate the performance overhead of Config-ECC, we inject additional read and write operations for memory requests of various granularities, as summarized in Section 6.5. We ran simulations by assuming that each 32B write always has a 32B read before it. This represents the

2. RATT-ECC is also an ECC scheme based on 64B access and it has similar performance/energy results compared to Citadel.

TABLE 4
Read/Write Overhead of Different ECC Schemes

	32B Read	32B Write	64B Read	64B Write
32B ECC [1]	32B read + CRC-16	32B write to update CRC-16	64B read + CRC-16 decoding	64B write to update CRC-16
64B ECC [13]	64B read + CRC-32	64B read + CRC-32 decoding 64B write to update CRC-32	64B read + CRC-32 decoding	64B write to update CRC-32
128B ECC	64B read + 128B ECC	128B read + 128B ECC decoding + 128B write to update 128B ECC	128B read + 128B ECC decoding	128B read + 128B write to update 128B ECC scheme
Config-ECC	32B read + CRC-24	32B read + CRC-24 decoding + 64B write to update CRC-24 + RS (72,70)	64B read + CRC-24 & RS (72,70) decoding	64B write to update CRC-24 & RS(72,70)

worst case performance scenario for Config-ECC. Our evaluation results show that the increase in the write latency has negligible performance impact across the diverse set of applications. The average performance degradation is 1.2 percent with the worst case of 3.7 percent for RD_SRA_2.

We attribute the insignificant performance penalty of Config-ECC to four major reasons. First, a write operation is not latency critical. The GPU pipeline does not need to stall for the completion of write operations. Second, GPU applications are less latency sensitive [45], [46], [47], [48]. The slight latency increase, incurred for 32B write operations, can be well overlapped by the multithreading execution feature of GPUs. Third, write operations, in particular 32B granularity accesses, account for a small fraction of all DRAM accesses for GPGPU applications. Our characterization results show the fraction of write operations ranges from 0.8 to 49.1 percent with an average of only 20.8 percent for all benchmarks. Finally, the latency to read or write 32B or 64B data in HBM2 is similar because of the pseudo-channel feature of HBM2. A single 64B read or write operation can be split into two 32B operations, which are performed concurrently. Consequently, the latency overhead is tolerable under Config-ECC. Overall, our performance evaluation shows that, compared with the baseline 32B ECC scheme, the execution time speedup is only degraded slightly by a negligible 1.2 percent on average for applications that prefer a smaller cache line granularity, as shown in Fig. 9.

64B ECC scheme and 128B ECC scheme have more additional reads and writes compared to Config-ECC for the benchmarks that prefer small cache line size. However, timing overhead does not lead to huge difference but the energy consumptions lead to large difference. Hence, we perform a comprehensive comparsion for DRAM energy in the later section.

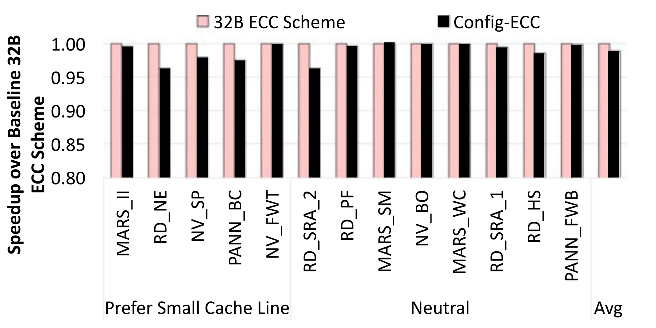


Fig. 9. Performance variation with Config-ECC (The performance of Config-ECC is evaluated under the worst case, such that all write operations are in 32B and cannot be coalesced into a 64B/128B operation).

7.2 Reliability Comparison

7.2.1 32B Access Schemes

We provide detailed comparison of our Config-ECC 32B scheme to the baseline CRC-16 scheme [1] and the single bit correction and double bit detection code (SEC-DED) used by Nvidia Tesla P100 [2]. Other schemes such as the rotational code which uses 16b ECC to protect 128b data [12] and BCH based schemes are not as competition. For instance, the scheme in [12] has an SDC rate of 3E-2, which is too high for detecting errors due to large granularity faults, which accounts for 36 percent of all faults in Table 7. Similiarly, a BCH code that can be used to correct 3 bit errors and maybe detect 4 bit errors (there is no BCH code to correct 4 bit errors with 32 ECC bits) also does not provide strong detection capability for errors due to large granularity faults. It is not even strong enough to correct 4 bit errors caused by a single TSV failure.

The SDC rate of Config-ECC 32B scheme is dominated by the detection capability of the CRC-24 code. The reason is that once CRC code miss-detects errors, the RS decoder is not activated. Simulation result shows that CRC-24 has SDC rate of $7E-8$ (10 billion runs), which is very close to its theoretical SDC rate $2^{-24} \cong 5.9E-8$.

Both the baseline CRC-16 scheme and the SEC-DED scheme have higher SDC rate. CRC-16 can detect all errors due to small granularity faults and has an SDC rate of $1.5E-5$ for errors due to large granularity faults. The disadvantage of using CRC-16 as the tier-1 code is that it can only perform detection and all correction has to be done by the tier-2 code.

We assume SEC-DED code in [2] is based on Hamming (72,64), which protects 64 data bits with 8 parity ECC bits. The Hamming (72,64) SEC-DED can correct all errors due to small granularity faults but it has very poor SDC rate for errors due to large granularity faults. The error performance of the three competing schemes is shown in Table 5.

Using raw FIT rate number from [11], we calculate the final FIT rates of all schemes. CRC-16 and SEC-DED have final FIT rate of 3.7E-3 and 68.9, respectively. In contrast, our Config-ECC 32B reduces the raw FIT rate to 1.69E-5, which is 200x better than the baseline CRC-16.

Compared to Citadel or RATT-ECC (64B ECC schemes), our CRC-24 has higher SDC rate (300x). Both Citadel and RATT-ECC designs are optimized for 64B access and thus force the system to read 64B even if the application only needs 32B. However, the product code design of Config-ECC allows the system to either choose better performance but lower reliablity(32B access with CRC-24) or better reliablity

TABLE 5
The Error Protection Coverage for 32B Config-ECC and the Existing Schemes

	CRC-16 (Baseline)	SEC-DED (Nvidia)	Config-ECC 32B (Proposed)
Single bit or column failure	DCE:0% DUE:100% SDC:0%	DCE:100% DUE:0% SDC:0%	DCE:100% DUE:0% SDC:0%
Single TSV failure	DCE:0% DUE:100% SDC:0%	DCE:100% DUE:0% SDC:0%	DCE:100% DUE:0% SDC:0%
Single row or bank failure	DCE:0% DUE:(1-1.5E-5) SDC:1.5E-5	DCE:0% DUE:72% SDC:28%	DCE:0% DUE:(1-7E-8) SDC:7E-8

(64B access with CRC - RS product code) for 32B access application. The next section explains why our tier-1 code has stronger reliablity compared to the current 64B ECC schemes.

7.2.2 64B Access Schemes

We compare Config-ECC 64B scheme to two recent ECC schemes: Citadel [13] and RATT-ECC [11]. For 64B access without interleaving, SDC events can occur due to (i) two CRC-24 decoders declaring error free, and RS (72,70) decoder also declaring error free in spite of errors, (ii) one CRC-24 decoder declaring errors, RS (72,70) performing correction and two CRC-24 decoder declaring error free even when errors are presented. The SDC rate of (ii) is very low because the probability of two CRC-24 codes detecting errors wrongly at the same time is extremely low (around $2^{-24} \times 2^{-24}$). We assume the worst condition when errors are all located in one 32B of data line in case (i). In this case, the SDC rate is calculated theoretically to $2^{-24} \times 2^{-16} \cong 9.0E-13$ (close to 10^{-12}). We did not see any error events in 10^{12} runs and so we conservatively claim that the SDC rate is lower than 10^{-12}. Note that the interleaving 64B access scheme used in static mode have similar SDC rate but stronger burst error detection capability.

Citadel [13] uses CRC-32 as the tier-1 code to detect errors and relies on tier-2 code to perform all correction. RATT-ECC [11] uses a stronger tier-1 code, RS(70,64), and so can correct all errors due to small granularity faults. The extended version of Citadel [49] uses CRC-30 (instead of CRC-32) and single bit correction, and has lower detection capability compared to the original scheme [13]. The

TABLE 6
The Error Protection Coverage for 64B Config-ECC and the Existing Schemes

	CRC-32 [13]	RS(70,64) [11]	Config-ECC 64B (Proposed)
Single bit or column failure	DCE:0% DUE:100% SDC:0%	DCE:100% DUE:0% SDC:0%	DCE:100% DUE:0% SDC:0%
Single TSV failure	DCE:0% DUE:100% SDC:0%	DCE:100% DUE:0% SDC:0%	DCE:100% DUE:0% SDC:0%
Single row or bank failure	DCE:0% DUE: 1 - SDC SDC:$2.3E-10$	DCE:0% DUE: 1 - SDC SDC: $2.4E-10$	DCE:0% DUE: $1-SDC$ SDC: $1E-12$

TABLE 7
The Final FIT Rate of Config-ECC Schemes

Failure Mode	Raw FIT	32B	64B/128B
Single bit	238	0	0
Single column	70	0	0
Single row	84	5.9E-6	8.4E-11
Single bank	162	1.1E-5	1.6E-10
Single TSV	41	0	0
Summary	685	1.69E-5	2.44E-10

simulation results show that our Config-ECC 64B scheme has SDC rate lower than $1E-12$, making it at least 40x stronger than [13] and [11]. Our Config-ECC 64B reduces the raw FIT rate to 2.44E-10, which is 20x better than [13] and [11], and 40x better than [49].

Parity-Helix [21] can protect from a whole channel failure or a whole die failure. It has a storage overhead of 14.3 percent for the tier-2 code and 26.8 percent for the overall ECC design (tier-1 + tier-2). While Config-ECC uses XOR-correction code to correct errors due to a single bank failure, it can be extended to support 3DP [13] or Parity-Helix [21].

7.2.3 128B Access Schemes

Since there is no current ECC scheme designed for 128B access, we only analyze the reliability metrics of our Config-ECC 128B scheme. In the dynamic mode, Config-ECC 128B is built using two Config-ECC 64B units and so the reliability performance of Config-ECC 128B is the same as Config-ECC 64B. In the static mode, Config-ECC 128B uses an interleaver to detect up to 96 consecutive burst errors. Table 7 summarizes the final FIT rates of our Config-ECC.

7.3 Energy Comparison

The energy overhead is due to the mismatch between the data access size and data size used for the different ECC schemes. It is a function of the numbers of reads, writes, and row activations.

7.3.1 Dynamic Operation Mode

We present the energy results of the four competing schemes in Fig. 10.[3] For the GPGPU applications that show a mixed data fetch size preference, RD_KMN, PAN_PAG, PAN_SPM, MAR_PVR, and RD_BFS (the first five applications in Fig. 1), Config-ECC achieves a good balance between strong reliability guarantee and low energy consumption. Compared to the lowest energy 32B ECC design, Config-ECC increases the active energy consumption by 1.5 percent while providing 200 times stronger reliability guarantee. The last five benchmarks, NV_BIO, RD_PF, PAN_CMX, RD_STC and PAN_MIS have small fractions of accesses that are 32B and 64B, so the energy results are also very similar among all the schemes.

3. We observe that, of the 15 benchmarks listed in Table 3, PAN_FW, RD_HOT, NV_MC, PAN_FB and RD_BP have mostly 128B accesses (see Fig. 1). Therefore, there is no difference in the energy performance of the four ECC schemes. Thus, we do not show their energy results in Fig. 10.

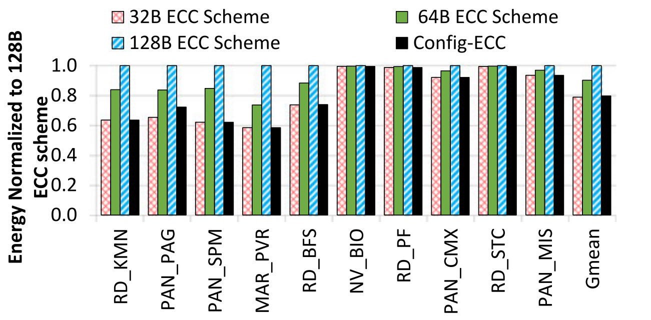


Fig. 10. Energy comparison of the competing schemes when operating in the dynamic operation mode.

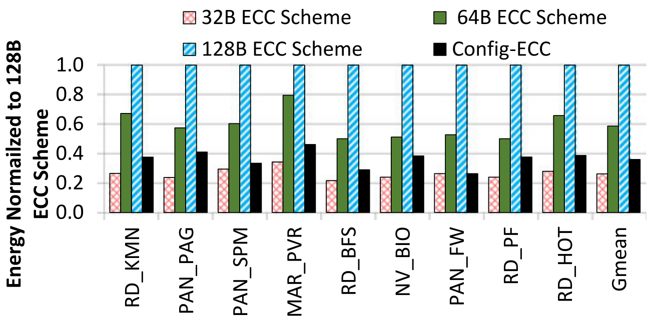


Fig. 11. Energy consumption of the competing schemes when operating in the static operation mode.

When compared to the 64B and 128B ECC designs, Config-ECC offers 20 times stronger reliability guarantee while requiring significantly lower active energy consumption by 17 and 34 percent less than [13] and the 128 ECC scheme. Both [13] and 128B ECC schemes have higher energy due to the mismatch between access size and ECC data size, resulting in larger number of reads, writes, and row activations. An analysis of the ratio of reads and writes shows that these benchmarks have high frequency of 32B reads but very low frequency of 32B writes. This is why Config-ECC has energy consumption similar to [1] even though it has higher energy consumption for 32B writes compared to [1].

7.3.2 *Static Operation Mode*

For the static operation mode, we present the energy consumption of the competing schemes for the L1 cache-off configuration, since the energy results for the L1 cache-on configuration are expected to be the same for the competing schemes. This corresponds to the first nine GPGPU applications in Fig. 3. In this configuration, the requests are either 32B read or write requests. Fig. 11 shows the energy components of Config-ECC and other ECC schemes. Config-ECC consumes an additional 10 percent energy compared to [1] since the number of 32B writes for these benchmarks ranges from 0 to 49 percent. This energy overhead comes with the benefit of a stronger, more reliable memory system. As what Section 7.2 shows, Config-ECC provides 200 times lower SDC rate than the 32B ECC scheme. When compared to [13] and 128B ECC scheme, Config-ECC has 21 and 63 percent lower energy respectively.

8 Conclusion

HBM is projected to boost performance and reduce power and energy in future data centers; however, its reliability is an issue. The design of ECC is challenging because future GPUs are expected to implement a memory subsystem supporting fine and coarse-grained data accesses to match the difference in the spatial locality of GPGPU applications. Given the differences in the cache/data line size, an ECC scheme that is optimized for a fixed data line size is sub-optimal.

We present Config-ECC, a two-tiered error correction scheme that provides high reliability with a flexible structure to support different sized accesses in HBM memory systems. The configurable/flexible tier-1 code is a product code that provides strong error detection and correction capabilities to correct all errors due to small granularity faults and detect errors with large granularity faults with very low silent data corruption (SDC) rate. The tier-2 code is launched for correcting errors due to large granularity faults. Config-ECC increases the reliability by 200 times compared to a fixed 32B ECC scheme and by 20 times compared to a fixed 64B ECC scheme with 1.2 percent performance degradation. Also, Config-ECC can choose to read only 32B resulting in 17 percent energy reduction when operating in dynamic mode and 21 percent reduction when operating in static mode compared to a fixed 64B ECC scheme.

Acknowledgments

The authors would like to thank the anonymous reviewers for their insightful feedback, which has been used to improve the paper. This work is supported in part by the US National Science Foundation under CCF-1618039 and DARPA PERFECT.

References

[1] *High Bandwidth Memory (HBM) DRAM*, JEDEC Standard, JESD235A, 2015.
[2] "Nvidia Tesla P100 - Whitepaper," *WP-08019–001 v01.1*, 2016.
[3] J. Macri, "AMD's next generation GPU and high bandwidth memory architecture: FURY," in *Proc. IEEE Hot Chips 27 Symp.*, Aug. 2015, pp. 1–26.
[4] S. Lin and D. J. Costello, *Error Control Coding*, 2nd ed. London, U.K.: Pearson, 2004.
[5] B. Jacob, S. Ng, and D. Wang, *Memory Systems Cache, DRAM, Disk*, 1st ed. San Mateo, CA, USA: Morgan Kaufmann, 2007.
[6] T. R. Rao and E. Fujiwara, *Error-Control Coding for Computer Systems*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1989.
[7] AMD, "White paper - AMD Graphics Cores Next (GCN) Architecture," Jun. 2012.
[8] M. Rhu, M. Sullivan, J. Leng, and M. Erez, "A locality-aware memory hierarchy for energy-efficient GPU architectures," in *Proc. 46th Annu. IEEE/ACM Int. Symp. Microarchit.*, 2013, pp. 86–98.
[9] A. Arunkumar, S.-Y. Lee, and C.-J. Wu, "ID-Cache: Instruction and memory divergence based cache management for GPUs," in *Proc. IEEE Int. Symp. Workload Characterization*, Sep. 2016, pp. 158–167.
[10] W. Jia, K. A. Shaw, and M. Martonosi, "Characterizing and improving the use of demand-fetched caches in GPUs," in *Proc. 26th ACM Int. Conf. Supercomput.*, 2012, pp. 15–24.
[11] H.-C. Chen, C.-J. Wu, T. Mudge, and C. Chakrabarti, "RATT-ECC: Rate adaptive two-tiered error correction codes for reliable 3D die-stacked memory," *ACM Trans. Archit. Code Optimization*, vol. 13, pp. 24:1–24:24, Sep. 2016.
[12] H. Jeon, G. Loh, and M. Annavaram, "Efficient RAS support for die-stacked DRAM," in *Proc. IEEE Int. Test Conf.*, Oct. 2014, pp. 1–10.
[13] P. Nair, D. Roberts, and M. Qureshi, "Citadel: Efficiently protecting stacked memory from large granularity failures," in *Proc. 47th Annu. IEEE/ACM Int. Symp. Microarchit.*, Dec. 2014, pp. 51–62.
[14] *High bAndwidth Memory (HBM) DRAM*, JEDEC Standard JESD235, 2013.

[15] K. Sohn, W. J. Yun, R. Oh, C. S. Oh, S. Y. Seo, M. S. Park, D. H. Shin, W. C. Jung, S. H. Shin, J. M. Ryu, H. S. Yu, J. H. Jung, K. W. Nam, S. K. Choi, J. W. Lee, U. Kang, Y. S. Sohn, J. H. Choi, C. W. Kim, S. J. Jang, and G. Y. Jin, "18.2 a 1.2 V 20 nm 307GB/s HBM DRAM with at-speed wafer-level I/O test scheme and adaptive refresh considering temperature distribution," in *Proc. IEEE Int. Solid-State Circuits Conf.*, Jan. 2016, pp. 316–317.

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

[17] A. Hwang, I. Stefanovici, and B. Schroeder, "Cosmic rays don't strike twice: Understanding the nature of DRAM errors and the implications for system design," *SIGARCH Comput. Archit. News*, vol. 40, pp. 111–122, Mar. 2012.

[18] V. Sridharan, J. Stearley, N. DeBardeleben, S. Blanchard, and S. Gurumurthi, "Feng Shui of supercomputer memory: Positional effects in DRAM and SRAM faults," in *Proc. Int. Conf. High Perform. Comput. Netw. Storage Anal.*, 2013, pp. 1–11.

[19] V. Sridharan, N. DeBardeleben, S. Blanchard, K. B. Ferreira, J. Stearley, J. Shalf, and S. Gurumurthi, "Memory errors in modern systems: The good, the bad, and the ugly," in *Proc. 20th Int. Conf. Archit. Support Program. Languages Operating Syst.*, 2015, pp. 297–310.

[20] J. Kim, M. Sullivan, and M. Erez, "Bamboo ECC: Strong, safe, and flexible codes for reliable computer memory," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit.*, 2015, pp. 101–112.

[21] X. Jian, V. Sridharan, and R. Kumar, "Parity Helix: Efficient protection for single-dimensional faults in multi-dimensional memory systems," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit.*, Mar. 2016, pp. 555–567.

[22] J. Park, J. Park, and S. Bhunia, "VL-ECC: Variable data-length error correction code for embedded memory in DSP applications," *IEEE Trans. Circuits Syst. II: Exp. Briefs*, vol. 61, no. 2, pp. 120–124, Feb. 2014.

[23] S.-L. Gong, M. Rhu, J. Kim, J. Chung, and M. Erez, "CLEAN-ECC: High reliability ECC for adaptive granularity memory system," in *Proc. 48th Int. Symp. Microarchit.*, 2015, pp. 611–622.

[24] D. H. Yoon, M. K. Jeong, M. Sullivan, and M. Erez, "The dynamic granularity memory system," in *Proc. 39th Annu. Int. Symp. Comput. Archit.*, Jun. 2012, pp. 548–560.

[25] NVIDIA, "NVIDIA's next generation CUDA compute architecture: Fermi," Sep. 2009, https://www.nvidia.com/content/PDF/fermi_white_papers/P.Glaskowsky_NVIDIA's_Fermi-The_First_Complete_GPU_Architecture.pdf

[26] Intel, "The compute architecture of Intel processor graphics Gen9," Aug. 2015, https://software.intel.com/sites/default/files/managed/c5/9a/The-Compute-Architecture-of-Intel-Processor-Graphics-Gen9-v1d0.pdf

[27] NVIDIA, "NVIDIA GeForce GTX 750 Ti: Featuring first-generation Maxwell GPU technology, designed for extreme performance per watt," Feb. 2014, https://international.download.nvidia.com/geforce-com/international/pdfs/GeForce-GTX-750-Ti-Whitepaper.pdf

[28] D. Yoon and M. Erez, "Virtualized ECC: Flexible reliability in main memory," *IEEE MICRO*, vol. 31, no. 1, pp. 11–19, Jan./Feb. 2011.

[29] A. N. Udipi, N. Muralimanohar, R. Balsubramonian, A. Davis, and N. P. Jouppi, "LOT-ECC: Localized and tiered reliability mechanisms for commodity memory systems," in *Proc. Int. Symp. Comput. Archit.*, Jun. 2012, pp. 285–296.

[30] X. Jian, H. Duwe, J. Sartori, V. Sridharan, and R. Kumar, "Low-power, low-storage-overhead Chipkill correct via multi-line error correction," in *Proc. Int. Conf. High Perform. Comput. Netw. Storage Anal.*, 2013, pp. 1–12.

[31] T. Kasami and S. Lin, "On the probability of undetected error for the maximum distance separable codes," *IEEE Trans. Commun.*, vol. 32, no. 9, pp. 998–1006, Sep. 1984.

[32] W. W. Peterson and D. T. Brown, "Cyclic codes for error detection," *Proc. IRE*, vol. 49, no. 1, pp. 228–235, Jan. 1961.

[33] P. Koopman, "Best CRC polynomials," https://users.ece.cmu.edu/~koopman/crc/

[34] P. Koopman and T. Chakravarty, "Cyclic redundancy code (CRC) polynomial selection for embedded networks," in *Proc. Int. Conf. Depend. Syst. Netw.*, Jun. 2004, pp. 145–154.

[35] J. J. Kong and K. K. Parhi, "Interleaved cyclic redundancy check (CRC) code," in *Proc. 37th Asilomar Conf. Signals Syst. Comput.*, Nov. 2003, pp. 2137–2141.

[36] M. Matsumoto and T. Nishimura, "Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator," *ACM Trans. Model. Comput. Simul.*, vol. 8, pp. 3–30, Jan. 1998.

[37] J. Sim, G. H. Loh, V. Sridharan, and M. O'Connor, "Resilient die-stacked dram caches," in *Proc. Annu. Int. Symp. Comput. Archit.*, 2013, pp. 416–427.

[38] NVIDIA, "CUDA C/C++ SDK code samples v4.0," May 2011.

[39] B. He, W. Fang, N. K. Govindaraju, Q. Luo, and T. Wang, "Mars: A MapReduce framework on graphics processors," in *Proc. 17th IEEE/ACM Int. Conf. Parallel Archit. Compilation Techn.*, Oct. 2008, pp. 260–269.

[40] S. Che, B. M. Beckmann, S. K. Reinhardt, and K. Skadron, "Pannotia: Understanding irregular GPGPU graph applications," in *Proc. IEEE Int. Symp. Workload Characterization*, Sep. 2013, pp. 185–195.

[41] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, S.-H. Lee, and K. Skadron, "Rodinia: A benchmark suite for heterogeneous computing," in *Proc. IEEE Int. Symp. Workload Characterization*, 2009, pp. 44–54.

[42] S. Che, J. W. Sheaffer, M. Boyer, L. G. Szafaryn, L. Wang, and K. Skadron, "A characterization of the Rodinia benchmark suite with comparison to contemporary CMP workloads," in *Proc. IEEE Int. Symp. Workload Characterization*, 2010, pp. 1–11.

[43] P. Rosenfeld, E. Cooper-Balis, and B. Jacob, "DRAMSim2: A cycle accurate memory system simulator," *IEEE Comput. Archit. Lett.*, vol. 10, no. 1, pp. 16–19, Jan.–Jun. 2011.

[44] K. Chen, S. Li, N. Muralimanohar, J. H. Ahn, J. B. Brockman, and N. P. Jouppi, "CACTI-3DD: Architecture-level modeling for 3D die-stacked DRAM main memory," in *Proc. Des. Autom. Test Eur. Conf. Exhib.*, Mar. 2012, pp. 33–38.

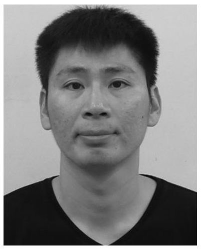
[45] J. Lee and H. Kim, "TAP: A TLP-aware cache management policy for a CPU-GPU heterogeneous architecture," in *Proc. 18th IEEE/ACM Int. Symp. High Perform. Comput. Archit.*, Feb. 2012, pp. 1–12.

[46] V. Mekkat, A. Holey, P.-C. Yew, and A. Zhai, "Managing shared last-level cache in a heterogeneous multicore processor," in *Proc. 22nd IEEE/ACM Int. Conf. Parallel Architect. Compilation Techn.*, Oct. 2013, pp. 225–234.

[47] J. Hestness, S. W. Keckler, and D. A. Wood, "A comparative analysis of microarchitecture effects on CPU and GPU memory system behavior," in *Proc. IEEE Int. Symp. Workload Characterization*, Oct. 2014, pp. 150–160.

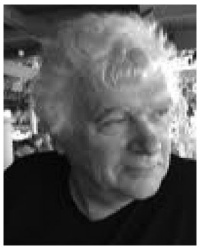
[48] O. Kayıran, N. C. Nachiappan, A. Jog, R. Ausavarungnirun, M. T. Kandemir, G. H. Loh, O. Mutlu, and C. R. Das, "Managing GPU concurrency in heterogeneous architectures," in *Proc. 47th Annu. IEEE/ACM Int. Symp. Microarchit.*, Dec. 2014, pp. 114–126.

[49] P. Nair, D. Roberts, and M. Qureshi, "Citadel: Efficiently protecting stacked memory from TSV and large granularity failures," *ACM Trans. Archit. Code Optimization*, vol. 12, pp. 49:1–49:24, Jan. 2016.



Hsing-Min Chen received the BS and MS degrees both in computer engineering from National Chiao Tung University, Taiwan, in 2007 and 2009, respectively, and the PhD degree in electrical engineering from Arizona State University, in 2017 spring. Currently, he is a senior CPU RAS (reliability, availability and serviceability) architecture engineer with Intel, Santa Clara, California. His research interests include DRAM/memory reliability, memory architecture and performance, and CPU RAS.

Shin-Ying Lee received the BS degree in electrical engineering from National Cheng Kung University, Taiwan, in 2008, the MS degree in computer engineering from National Cheng Kung University, Taiwan, in 2010, and the PhD degree in computer engineering from Arizona State University, in 2017. He is now focusing on SoC architecture and performance modeling with Samsung Austin R&D Center. His research interests include high performance processor architecture, memory hierarchy design, as well as operating system optimization.



Trevor Mudge received the PhD degree in computer science from the University of Illinois, Urbana. He is now the Bredt Family professor of computer science for pioneering contributions to low-power computer architecture and received the University of Illinois Distinguished Alumni Award. He is a life fellow of the IEEE, a member of the ACM, the IET, and the British Computer Society.

Carole-Jean Wu received the BS degree in electrical and computer engineering from Cornell University, in 2006, and the MA and PhD degrees in electrical engineering from Princeton University, in 2008 and 2012, respectively. She is currently an assistant professor with the School of Computing, Informatics, and Decision Systems Engineering, Arizona State University. Her research interests include the areas of processor architectures and memory hierarchy designs to achieve high performance and Improved energy efficiency. She is a senior member of the IEEE.

Chaitali Chakrabarti received the BTech degree in electronics and electrical communication engineering from the Indian Institute of Technology, Kharagpur, India, in 1984, and the PhD degree in electrical engineering from the University of Maryland, College Park, in 1990. She is a professor with the School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe. Her research interests include VLSI algorithm-architecture co-design of signal processing and communication systems and all aspects of low-power embedded systems design. She is a fellow of the IEEE.