Stealth ECC: A Data-Width Aware Adaptive ECC Scheme for DRAM Error Resilience

Young Seo Lee[1], Gunjae Koo[1], Young-Ho Gong[2], and Sung Woo Chung[1]
[1]Department of Computer Science and Engineering, Korea University, Seoul 02841, South Korea
[2]School of Computer and Information Engineering, Kwangwoon University, Seoul 01897, South Korea
E-mail: {leeyoungseo, gunjaekoo, swchung}@korea.ac.kr; yhgong@kw.ac.kr

***Abstract*—As DRAM process technology scales down and DRAM density continues to grow, DRAM errors have become a primary concern in modern data centers. Typically, data centers have adopted memory systems with a single error correction double error detection (SECDED) code. However, the SECDED code is not sufficient to satisfy DRAM reliability demands as memory systems get more vulnerable. Though the servers in data centers employ strong ECC schemes, such ECC schemes lead to substantial performance and/or storage overhead.**

In this paper, we propose *Stealth ECC*, a cost-effective memory protection scheme providing stronger error correctability than the conventional SECDED code, with negligible performance overhead and without storage overhead. Depending on the data-width (either narrow-width or full-width), Stealth ECC adaptively selects ECC schemes. For narrow-width values, Stealth ECC provides multi-bit error correctability by storing more parity bits in MSB side, instead of zeros. Furthermore, with bitwise interleaved data placement between x4 DRAM chips, Stealth ECC is robust to a single DRAM chip error for narrow-width values. On the other hand, for full-width values, Stealth ECC adopts the SECDED code, which maintains DRAM reliability comparable to the conventional SECDED code. As a result, thanks to the reliability improvement of narrow-width values, Stealth ECC enhances overall DRAM reliability, while incurring negligible performance overhead as well as no storage overhead. Our simulation results show that Stealth ECC reduces the probability of system failure (caused by DRAM errors) by 47.9%, on average, with only 0.9% performance overhead compared to the conventional SECDED code.

Keywords*—*DRAM reliability, narrow-width value, error correction code, chip error resilience

I. Introduction

The criticality of DRAM reliability grows fast as DRAM process technology scales down and DRAM density continues to grow. Though DRAM process technology scaling enables higher performance and density of DRAMs, it makes DRAM cells more vulnerable to manufacturing variation and imperfection [8][9][12], which in turn brings severe DRAM reliability problems. To meet DRAM reliability demands, memory systems typically employ error correction code (ECC) dual in-line memory modules (DIMMs) in data centers. Among various ECC schemes, a single error correction double error detection (SECDED) code is the most widely exploited ECC scheme, which has 8-bit parity for every 64-bit data. However, as the likelihood of multi-bit errors increases due to high permanent error rate in DRAM [6], the conventional SECDED code is not sufficient to maintain DRAM reliability demands.

To tackle this problem, researchers have proposed stronger ECC schemes than the conventional SECDED code [5][17][18]. However, such ECC schemes cause substantial performance and/or storage overhead. For example, a commercial *Chipkill* corrects any single DRAM chip error by exploiting a symbol-based ECC with two channels of ECC DIMMs [18]. Though *Chipkill* provides DRAM chip error resilience, it is necessary to simultaneously activate two channels per a single memory request. Thus, *Chipkill* degrades channel-level parallelism and effective memory bandwidth, causing up to 25% performance overhead [5]. LOT-ECC reaches near *Chipkill*-level reliability with a single channel of ECC DIMMs, based on the multi-tier error correction [17]. However, it requires 12.4% storage overhead and performance overhead similar to the commercial *Chipkill*. In addition, CARE tries to mitigate the performance overhead while providing near *Chipkill*-level reliability by employing a simple 6-bit correctable ECC scheme with operating system (OS) support [5]. Nevertheless, it still incurs 10% performance overhead.

In this paper, we propose *Stealth ECC*, a cost-effective memory protection scheme providing stronger error correctability than the conventional SECDED code. To provide stronger error correctability, Stealth ECC exploits unmeaningful data fields frequently observed in DRAM. Our motivation experiment reveals that a large portion of data words are narrow-width values which contain meaningful data values in the least significant bits (LSBs) while the remaining most significant bits (MSBs) are filled with zeros. Depending on the data-width (either narrow-width or full-width), Stealth ECC adaptively selects ECC schemes. For narrow-width values, Stealth ECC stores more parity bits in MSB side, replacing zeros, which in turn provides multi-bit correctability. Furthermore, with bitwise interleaved data placement between x4 DRAM chips, Stealth ECC is robust to a single DRAM chip error for narrow-width values. On the other hand, for full-width values, Stealth ECC adopts the SECDED code, which maintains DRAM reliability comparable to the conventional SECDED code. Since the additional parity bits are stored in the zero parts of narrow-width values, it does not require any additional storage overhead. In addition, the multi-bit error correction of Stealth ECC has light-weight computation for parity check, where the decoding latency is only a few cycles. Consequently, with the error mitigation for narrow-width values, Stealth ECC enhances overall DRAM reliability while causing negligible performance overhead as well as no storage overhead compared to the conventional SECDED code.

II. Background and Motivation

With stronger ECC schemes, memory systems get more robust to errors at the cost of storage and/or performance overhead. We consider the Bose-Chaudhuri-Hocquenghem (BCH) code, which is a commonly employed ECC in modern memory systems due to its light-weight parity check computation. Note BCH(n,k,t) provides t-bit correctability and ($t+1$)-bit detectability for a k-bit data word by exploiting a (n-k)-bit parity. The encoding process for BCH(n,k,t) converts an original k-bit data word to a n-bit codeword. For example, BCH(72,64,1) provides 1-bit correctability and 2-bit detectability for the 64-bit data word by exploiting the 8-bit parity, storing the 72-bit codeword to DRAM chips.

We quantitatively investigate the overhead of stronger BCH codes (i.e., multi-bit correctable BCH codes), compared to the conventional SECDED code (i.e., BCH(72,64,1)); the

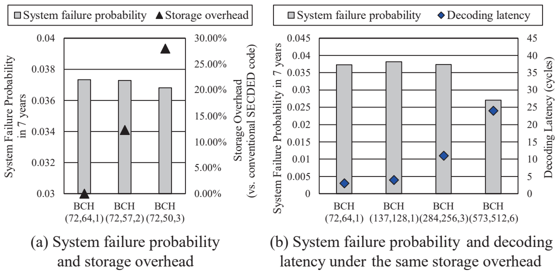

Fig. 1. System failure probability and overhead across different BCH codes.

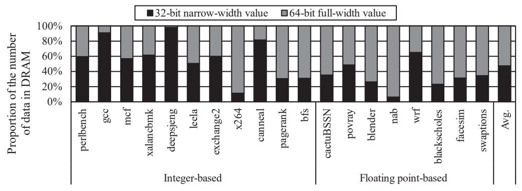

Fig. 2. Proportion of 32-bit narrow-width values in DRAM.

detailed simulation environment will be described in Section IV. We analyze the storage overhead across different BCH codes with 72-bit codeword size; we fix the codeword size to 72-bit, considering a single memory burst length (72-bit) of standard ECC DIMMs. As shown in Fig. 1(a), the system failure probability is not much reduced at the substantial cost of storage overhead; the storage overhead is exponentially proportional to the number of correctable bits. For example, though the 3-bit correctable BCH code (i.e., BCH(72,50,3)) reduces 1.4% system failure probability, it causes additional storage overhead by 28% compared to the conventional SECDED code.

To provide strong error correctability without such storage overhead, the BCH code should be adopted for multiple memory bursts rather than a single memory burst. However, the BCH code with multiple memory bursts increases decoding latency, which may degrade system performance. Fig. 1(b) shows the system failure probability and decoding latency with respect to the codeword size. We select the BCH codes with the highest possible correctability under the same storage overhead of the conventional SECDED code (i.e., $n/k \leq 72/64$). As the codeword size and the number of correctable bits increase, the system failure probability is reduced. However, the decoding latency of the BCH code is exponentially increased, potentially leading to substantial performance overhead. According to the preliminary experiments, it may cause storage and/or performance overhead to employ stronger ECC schemes.

To mitigate the storage and/or performance overhead while providing stronger error correctability, it would be beneficial to apply a stronger ECC scheme to meaningful data bits only. According to [15], 32-bit narrow-width values account for average 85.4% in the data cache when running the SPEC CPU 2017 benchmark suite [4] in 64-bit architectures. We investigate the proportion of 32-bit narrow-width values in DRAM for various workloads from several benchmark suites (SPEC CPU 2017 [4], PARSEC [2], and GAP [1]). As shown in Fig. 2, the proportion of 32-bit narrow-width values in DRAM accounts for 47.4%, on average. Thus, it is possible to exploit a considerable portion (i.e., zero parts) in DRAM to store more parity bits for meaningful data only, which improves DRAM reliability without storage overhead. In this paper, we exploit the narrow-width feature to improve DRAM

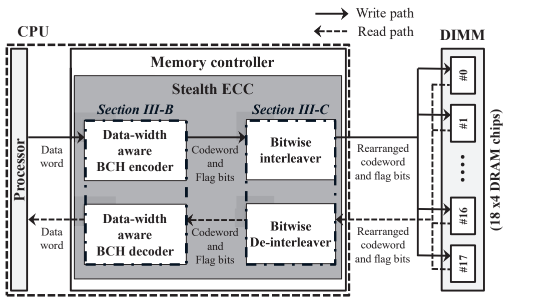

Fig. 3. Overview of the proposed scheme.

reliability with negligible performance overhead and without storage overhead.

III. Stealth ECC: Data-Width Aware Adaptive ECC

We propose *Stealth ECC*, a novel memory protection scheme which adaptively selects BCH codes depending on the data-width (either narrow-width or full-width). Stealth ECC mitigates system failure probability by exploiting a stronger BCH code for narrow-width values, storing more parity bits in MSB side, instead of zeros. At the same time, for full-width values, Stealth ECC provides 1-bit correctability and 2-bit detectability (i.e., SECDED) which is comparable to the conventional SECDED code.

A. Overview

As shown in Fig. 3, Stealth ECC requires a data-width aware BCH encoder/decoder and a bitwise interleaver/de-interleaver in the memory controller of a CPU. When storing a 64-bit data word to main memory, the data-width aware BCH encoder classifies the data word as either 32-bit narrow-width or 64-bit full-width. When the data word is narrow-width, the 32 LSBs (i.e., meaningful data) of the data word are encoded by the multi-bit correctable BCH code. Otherwise, the data word is encoded by the SECDED code. Since data words are encoded by different BCH codes depending on the data-width, a codeword should also be decoded differently depending on the data-width. Thus, the encoder generates flag bits to indicate the data-width (narrow-width or full-width) of a codeword; the detailed data-width aware adaptive BCH codes will be explained in Section III-B. Then, the bitwise interleaver rearranges both the codeword and flag bits; the detailed bitwise data placement will be explained in Section III-C. Both rearranged codeword and flag bits are stored to DRAM chips. To read data words from main memory, the bitwise de-interleaver loads the interleaved codeword and flag bits from DRAM chips and then restores them to original codeword and flag bits. Depending on the flag bits, the codeword is decoded by the corresponding BCH code in the data-width aware BCH decoder. Lastly, the decoded data word (i.e., original data word) is transferred to the processor.

B. Data-Width Aware ECC Selection

The data-width aware BCH encoder/decoder adaptively executes the encoding/decoding process depending on the data-width (either narrow-width or full-width). Accordingly, there needs a flag to distinguish the data-width, which in turn requires additional storage overhead. To avoid this storage overhead caused by the flag, we embed the *F* and *N* flags in the 144-bit corresponding to two memory bursts of standard ECC DIMMs. We exploit the flags to classify the two consecutive 64-bit data words (to be stored in DRAM through two memory bursts) into four cases depending on the data-

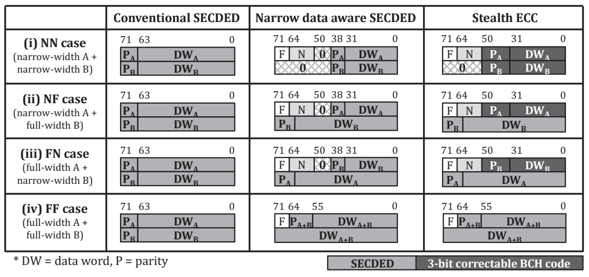
Conventional SECDED
Narrow data aware SECDED
Stealth ECC
(i) NN case (narrow-width A + narrow-width B)
(ii) NF case (narrow-width A + full-width B)
(iii) FN case (full-width A + narrow-width B)
(iv) FF case (full-width A + full-width B)
71 63 0
71 64 50 38 31 0
71 64 50 31 0
71 64 55 0
F
N
0
P_A
P_B
DW_A
DW_B
P_{A+B}
DW_{A+B}
* DW = data word, P = parity
SECDED
3-bit correctable BCH code

Fig. 4. Data word and parity layout corresponding to two memory bursts. (*F and N indicate the flags*)

width, as shown in Fig. 4; note the classification for two memory bursts does not require additional memory accesses, since a single memory access command generally requests eight memory bursts to DRAM chips considering the 64-byte cache block size. When the two consecutive data words contain at least one 32-bit narrow-width value (i.e., (i) NN, (ii) NF, and (iii) FN cases), we store the *F* flag in the zero part of the narrow-width value. Note the *F* flag indicates whether the two data words include at least one narrow-width value or not. However, when we consider the conventional SECDED code for each data word, the two consecutive full-width values (i.e., (iv) FF case) cannot have space to store the *F* flag. In this case, we apply the SECDED code to two data words at once, instead of applying the conventional SECDED code to each data word. While the SECDED code for each of two data words requires 16-bit (=8+8) parity, the SECDED code for two data words (128-bit) requires only 9-bit parity. With the SECDED code for two data words, it is possible to obtain 7-bit free space in the 144-bit (corresponding to two memory bursts). We store the *F* flag into this free space. In addition, we store the *N* flag into the remaining zero part of the narrow-width value to classify three cases (i.e., (i) NN, (ii) NF, and (iii) FN cases). Note the *N* flag indicates which data word is narrow-width in the two data words.

Based on the flags, it is possible to identify the data-width of each data word. When a data word is narrow-width, we can apply the SECDED code with meaningful data bits only, while leaving the zero parts; we call it *narrow data aware SECDED*. To further improve DRAM reliability, we are able to exploit the zero parts to store more parity bits for the stronger BCH code than the SECDED code, which corresponds to our proposed Stealth ECC. For narrow-width values in (i) NN, (ii) NF, and (iii) FN cases, the narrow data aware SECDED does not care about the errors in the zero parts, which improves DRAM reliability without any additional storage overhead. Though the narrow data aware SECDED exploits some of the zeros in MSB side to store *N* and *F* flags, there still remain zeros (e.g., 12-bit zeros in NF/FN case) in MSB side. Contrary to the narrow data aware SECDED, as shown in Fig. 4, Stealth ECC employs the 3-bit correctable BCH code by storing 12-bit additional parity into the zero part of a narrow-width value. Since Stealth ECC exploits the zero parts of narrow-width values, it does not cause any additional storage overhead, compared to the conventional SECDED code. In addition, the 3-bit correctable BCH code has light-weight computation for parity check, where the decoding latency is only a few cycles. On the other hand, for full-width values in the (ii) NF and (iii) FN cases, we apply the conventional SECDED code to each full-width value. In (iv) FF case, we apply the SECDED code to two full-width values at once, instead of applying the conventional SECDED code to each full-width value, to obtain the free space for the *F* flag. The SECDED code with the increased data word size may slightly degrade DRAM

TABLE I. REDUNDANT BITS OF FLAGS FOR ERROR TOLERABILITY

Flag	NN case	NF case	FN case	FF case
F flag (7-bit)	0000000	0000000	0000000	1111111
N flag (14-bit)	0000000 0000000	1111111 1111111	1111111 0000000	N/A

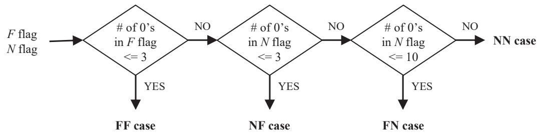

Fig. 5. Flag decoding process in the data-width aware BCH decoder.

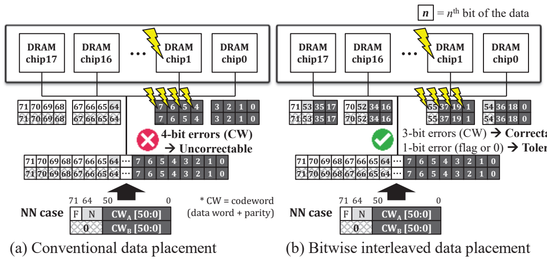
n = nth bit of the data
DRAM chip17
DRAM chip16
DRAM chip1
DRAM chip0
4-bit errors (CW) → Uncorrectable
3-bit errors (CW) → Correctable
1-bit error (flag or 0) → Tolerable
NN case
* CW = codeword (data word + parity)
CW_A [50:0]
CW_B [50:0]
(a) Conventional data placement
(b) Bitwise interleaved data placement

Fig. 6. Bit mapping depending on data placement methods in NN case.

reliability due to the reduced coverage of the SECDED code. However, even with the increased data word size, the reliability degradation of the SECDED code is not that significant as described in Fig. 1(b); BCH(137,128,1) leads to only 2.3% reliability degradation compared to the conventional SECDED code (i.e., BCH(72,64,1)). More importantly, since Stealth ECC improves error correction capability for narrow-width values significantly, it could improve overall DRAM reliability, which will be described in Section IV.

Though Stealth ECC improves DRAM reliability by applying the strong ECC scheme to narrow-width values, when errors occur in the flags, it causes system failure by misclassifying the data-width. To resolve this problem, we redundantly store *F* and *N* flags into 7 bits and 14 bits, respectively, as shown in Table I. With redundant bits of the flags, Stealth ECC achieves up to 3-bit tolerability for *F* and *N* flags, both. Fig. 5 depicts the flag decoding process in the data-width aware BCH decoder, to classify four cases (i.e., NN, NF, FN, and FF cases). By counting the number of zeros in each flag, Stealth ECC correctly classifies the data-width of two consecutive data words, even with up to 3-bit errors in each flag. For example, though the *F* flag is changed from 1111111 to 1011001 due to 3-bit errors, Stealth ECC still classifies this case as FF case based on the flag decoding process.

C. Bitwise Interleaving for Further Improving Reliability

For Stealth ECC, the bitwise interleaver rearranges the codeword and flag bits. Fig. 6 describes the bit mapping of Stealth ECC with the conventional and bitwise interleaved data placement in the standard ECC DIMMs, which consist of eighteen x4 DRAM chips. In Fig. 6, we depict the NN case as an example to show how the bitwise interleaved data placement prevents system failures from a single DRAM chip error for narrow-width values. As shown in Fig. 6(a), with the conventional data placement, each of four consecutive data bits in the 72-bit data is stored into a single DRAM chip. In this case, a single chip error leads to up to 4-bit errors for the codeword or flags, which are uncorrectable by the 3-bit correctable BCH code or intolerable by the redundant bits of the flags, respectively, thereby resulting in potential system

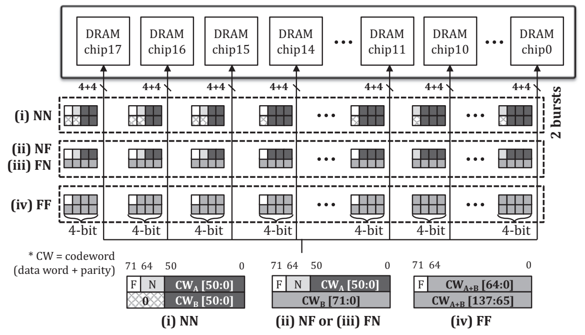

Fig. 7. Bitwise interleaved data placement of Stealth ECC.

failure. On the other hand, with the bitwise interleaved data placement, consecutive data bits are distributed into different DRAM chips; note the bitwise interleaved data placement is still capable of deploying the critical word first policy, since a single data word is loaded through one memory burst (not multiple memory bursts). As shown in Fig. 6(b), each of the first eighteen consecutive (i.e., 0th to 17th) data bits is stored into the DRAM chip0 to chip17, respectively, and then the following eighteen consecutive (i.e., 18th to 35th) data bits are stored in the same manner, and so on. Accordingly, any DRAM chip does not have more than 3 bits of a codeword. Though a single chip error occurs in the codeword, the 3-bit correctable BCH code is capable of correcting error bits. In addition, due to the redundant bits of the flags, each flag is tolerable to 3-bit errors. As a result, for narrow-width values, Stealth ECC endures a single chip error based on the bitwise interleaved data placement.

Fig. 7 shows all cases (i.e., NN, NF, FN, and FF cases) with the bitwise interleaved data placement of Stealth ECC. As shown in Fig. 7, the NF and FN cases have the same layout for narrow-width values with the NN case (i.e., 21-bit flags and 51-bit codeword). Therefore, as described in the above paragraph, the bitwise interleaved data placement further improves reliability for all the narrow-width values based on the 3-bit correctable BCH code and the redundant bits of the flags. On the other hand, for full-width values, the bitwise interleaved data placement does not further improve (nor degrade) reliability, compared to the conventional data placement. Consequently, the bitwise interleaved data placement improves overall DRAM reliability.

IV. Evaluation

A. Experimental Setup

We evaluate the DRAM reliability and performance of Stealth ECC, compared to the conventional SECDED code (baseline), narrow data aware SECDED (as described in Section III-B), and multi-bit correctable BCH codes. We select nineteen workloads with different proportion of narrow-width values, from widely exploited CPU benchmark suites (SPEC [4] and PARSEC [2]) and graph processing benchmark suite (GAP [1]). In case of the SPEC and PARSEC workloads, we fast-forward 10 billion instructions and then actually

TABLE II. Configurations for Performance Simulation

Parameter	Configuration
Processor	DerivO3CPU@3.0GHz
Cache	L1D/L1I: Private; 32KB; 8-way; 64B block L2: Private; 256KB; 4-way; 64B block
Memory controller	ECC decoding latency (cycle): Baseline (3), BCH(72,57,2) (7), BCH(72,50,3) (10), BCH(573,512,6) (24), narrow data aware SECDED (3), Stealth ECC (9)
Main memory	DDR4-2400; x4 bus-width; 18 chips; 16GB

execute 1 billion instructions. In case of the GAP workloads, we execute all instructions, using a bitcoin trust network as an input from Stanford network analysis platform (SNAP) [10].

We evaluate the system failure probability (caused by DRAM errors) using FaultSim [13], a memory reliability simulator exploiting real-world statistics of transient and permanent DRAM errors. We perform Monte-Carlo simulations on the system failure probability in 7 years with 1 million iterations, considering the typical operational lifespan of DRAM chips in servers [8]. For performance analysis, we estimate the performance penalty of each ECC scheme. The ECC decoding latency of each ECC scheme is calculated based on Strukov's model [16]. For additional latency of Stealth ECC, we implement the hardware components for Stealth ECC (shown in Fig. 3) in Verilog HDL and then synthesize them using Design Compiler based on SAED 14nm FinFET process technology [11]. According to the implementation results, it takes only one cycle to execute flag decoding (explained in Section III-B) and bitwise interleaving (explained in Section III-C) processes. Reflecting the performance penalties of ECC schemes, we evaluate the execution time of various workloads using the gem5 simulator [3]. The detailed system configuration for performance evaluation is shown in Table II. Furthermore, we analyze additional area and power overhead of Stealth ECC based on the implementation results.

B. Reliability Improvement

Fig. 8 depicts the system failure probability (caused by DRAM errors) depending on ECC schemes across various workloads. Stealth ECC reduces the average system failure probability by 47.9% and 26.6%, compared to the baseline and BCH(573,512,6), respectively, due to the following reasons. First, Stealth ECC provides up to 3-bit correctability (for narrow-width values) per a data word, while the baseline has only 1-bit correctability. In addition, Stealth ECC has stronger error correction capability than BCH(573,512,6), since BCH(573,512,6) has 6-bit correctability per eight data words (512-bit); Stealth ECC can correct up to 24-bit (=3-bit * 8) errors per eight data words. Second, Stealth ECC is robust to a single DRAM chip error for narrow-width values based on the bitwise interleaved data placement. Accordingly, as the proportion of narrow-width values increases, Stealth ECC further reduces system failure probability. Meanwhile, the narrow data aware SECDED reduces the system failure probability by average 22.6% compared to the baseline, since it does not care about the errors of zeros in MSB side, for

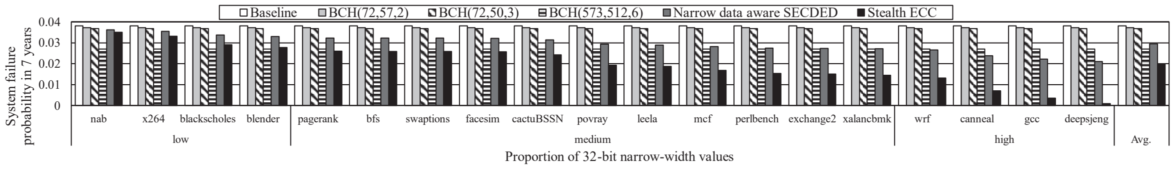

Fig. 8. System failure probability depending on ECC schemes across workloads. (*we categorize the workloads as low (~30%), medium (30~60%), or high (60%~) proportion of the narrow-width values*)

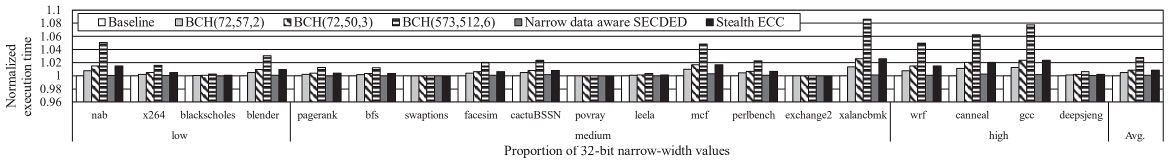

Fig. 9. Execution time depending on ECC schemes across workloads. (*we categorize the workloads as low (~30%), medium (30~60%), or high (60%~) proportion of the narrow-width values*)

narrow-width values. However, Stealth ECC always outperforms the narrow data aware SECDED, since it enhances error correctability by exploiting the zero parts of narrow-width values to store more parity bits; the narrow data aware SECDED is still vulnerable to multi-bit errors in the codeword (not zero parts).

With high proportion of narrow-width values (rightmost in Fig. 8), Stealth ECC significantly reduces the system failure probability, compared to the other multi-bit correctable BCH codes, thanks to the substantial error mitigation for narrow-width values. The narrow data aware SECDED also shows lower system failure probability than the other multi-bit correctable BCH codes, though it only provides a single bit error correctability. With medium proportion of narrow-width values (middle in Fig. 8), Stealth ECC prevents system failures from a single DRAM chip error for a considerable portion of narrow-width values, so that it is still effective for the overall reliability improvement than the others; the reliability improvement of Stealth ECC for narrow-width values is much higher than the reliability improvement of BCH(573,512,6). However, as the proportion of narrow-width values decreases, the narrow data aware SECDED is not effective for the reliability improvement than BCH(573,512,6), resulting in higher system failure probability. With low proportion of narrow-width values (leftmost in Fig. 8), Stealth ECC (as well as narrow data aware SECDED) still shows lower system failure probability than the baseline but higher system failure probability than BCH(573,512,6). Note BCH(573,512,6) has much longer decoding latency (2.4× compared to Stealth ECC).

C. *Performance Impact*

Fig. 9 shows the performance impact depending on the ECC schemes across various workloads. Stealth ECC shows the performance degradation by 0.9%, on average, compared to the baseline. For narrow-width values, the 3-bit correctable BCH code has light-weight computation for parity check, which requires 6 additional cycles per memory access for ECC decoding. Moreover, it requires 1 additional cycle for flag decoding and bitwise interleaving processes. Thus, Stealth ECC requires only 7 additional cycles per memory access of hundreds of cycles, leading to negligible performance overhead compared to the baseline; the baseline requires 3 cycles for ECC decoding. On the other hand, BCH(573,512,6) requires 24 cycles for ECC decoding latency, which causes performance overhead by up to 8.6% (2.8%, on average).

Meanwhile, the narrow data aware SECDED shows the performance degradation by 0.3%, on average, compared to the baseline. Since the narrow data aware SECDED adopts the 1-bit correctable BCH code with the conventional data placement, it does not require the additional latency for ECC decoding and bitwise interleaving. It only takes one additional cycle to classify the data-width as either narrow-width or full-width, resulting in negligible performance overhead.

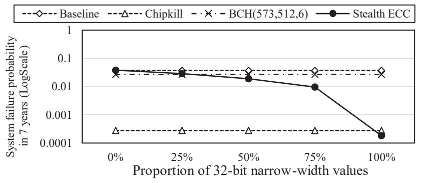

Fig. 10. Reliability depending on the proportion of narrow-width values.

D. *Area and Power Overhead*

According to the implementation results for Stealth ECC, the area is only 0.01mm^2. Since the memory controller area of a state-of-the-art server CPU is about 6.9mm^2 [19], the hardware components for Stealth ECC cause negligible area overhead (i.e., 0.1% of the memory controller area). In addition, the power consumption is only 1.63mW, which is negligible power overhead compared to the thermal design power (i.e., 205W) of the server CPU [19].

E. *Reliability depending on Narrow-width Value Ratio*

To verify the effectiveness of Stealth ECC, we evaluate the sensitivity of narrow-width value proportion to DRAM reliability. Fig. 10 shows the system failure probability of four different ECC schemes depending on the proportion of 32-bit narrow-width values. Due to the strong correction capability for narrow-width values, Stealth ECC increasingly reduces system failure probability as the proportion of narrow-width values increases, while the others have no changes in system failure probability. Only in case that the proportion of narrow-width values is extremely low (< 2.3%), Stealth ECC would cause marginal reliability degradation (2.2% even in case of no narrow-width value) compared to the baseline (while causing average 0.9% performance overhead). However, as shown in Fig. 2, even floating point-based workloads have 6.3~64.9% (33.7%, on average) proportion of narrow-width values. Accordingly, Stealth ECC is expected to provide better reliability than the baseline for most real-world workloads. In addition, when the proportion of narrow-width values is higher than 29.3%, Stealth ECC shows lower system failure probability than BCH(573,512,6). Moreover, Stealth ECC could be even better than *Chipkill* in an ideal case (i.e., the proportion of narrow-width values > 99.7%).

V. Related Work

Many previous studies have presented strong ECC schemes providing DRAM chip error resilience (i.e., *Chipkill*-level and near *Chipkill*-level ECC schemes). Gong et al. proposed CLEAN-ECC which exploits a two-tier ECC with adaptive memory access granularity [6]. CLEAN-ECC provides *Chipkill*-level reliability with course-grained memory accesses by adopting the symbol-based Reed-Solomon (RS) code, while enabling fine-grained memory access to error-free memory locations. Kim et al. also proposed Bamboo ECC employing the symbol-based RS code with cache block granularity, which provides *Chipkill*-level reliability [9]. Though CLEAN-ECC [6] and Bamboo ECC [9] provide *Chipkill*-level reliability, they cause performance overhead by average 19.3% and 20.3%, respectively, compared to the conventional SECDED code, due to the

TABLE III. SUMMARY OF RELATED WORKS AND STEALTH ECC

ECC scheme	Chipkill-level		Near Chipkill-level		
	CLEAN-ECC [6]	*Bamboo ECC [9]*	*LOT-ECC [17]*	*CARE [5]*	*Stealth ECC*
Storage overhead (vs. SECDED)	0%	0%	12.4%	0%[a]	0%
Avg. perf. overhead (vs. SECDED)	19.3%	20.3%	14.3%	10.0%	0.9%
OS support	No	No	No	Yes	No
Access granularity	16-64B	64B	64B	64B	16B
Scalability	Yes	Yes	Yes	No	Yes

[a] It additionally requires a 56KB ECC cache per 8GB DRAM.

expensive computation of the RS code. Udipi et al. presented LOT-ECC which exploits a four-tier hierarchical ECC consisting of a 1-tier local error detection code and 3-tier global error correction codes [17], reaching near *Chipkill*-level reliability. However, LOT-ECC causes 12.4% storage overhead as well as average 14.3% performance overhead compared to the conventional SECDED code. In addition, Chen et al. proposed CARE, a memory protection scheme, motivated by the observation that a correctable error is typically translated into an uncorrectable error later [5]. CARE stores the parity bits for a strong ECC in the cache-like structure (i.e., ECC cache), to avoid the storage overhead in DRAM. When multi-bit errors occur in a memory page, which is correctable by the strong ECC, CARE corrects the errors exploiting its ECC cache, and then retires the page based on the OS support, considering the occurrence of multi-bit errors as a warning sign of uncorrectable errors later. Though CARE improves DRAM reliability, it still causes average 10.0% performance overhead due to the page retirement, compared to the conventional SECDED code. Moreover, CARE requires increasingly larger ECC cache in the memory controller as the DRAM capacity increases, causing scalability problems. We summarize the characteristics of above strong ECC schemes compared to those of Stealth ECC, as shown in Table III; note our simulation environments described in Section IV-A are similar to those of strong ECC schemes. Different from the ECC schemes, Stealth ECC achieves DRAM chip error resilience for narrow-width values with i) no storage overhead, ii) negligible performance overhead, and iii) no OS support, still providing iv) fine-grained memory access and scalability. Due to the negligible storage and performance overhead, Stealth ECC can also be applied with manufacturing fault tolerance schemes such as ArchShield [12].

There also have been many studies exploiting the data compression techniques to make room for parity bits in a non-ECC DIMM [7][14]. Palframan et al. presented a memory protection scheme based on the block-level compression technique, which compresses a 64B cache block to 60B and then stores 4B parity bits [14]. They apply the SECDED code to each 15B sub-block by exploiting 1B parity. Hong et al. also introduced a memory protection scheme based on the two-dimensional zero compression technique, which distinguishes 8B non-zero data words by checking horizontal and vertical directions in a 64B cache block [7]. They adaptively apply the SECDED code or the 2-bit correctable BCH code to each 8B non-zero data word, depending on the compression ratio. However, the above mentioned studies have only 1-bit or 2-bit correctability and they are not tolerable to any single DRAM chip error, even with the ECC DIMM. On the contrary, Stealth ECC provides multi-bit correctability by storing additional parity bits in the zero parts of narrow-width values and it also deploys the bitwise interleaved data placement, which in turn achieves DRAM chip error resilience for narrow-width values.

VI. CONCLUSION

We propose Stealth ECC, a cost-effective memory protection scheme which adaptively applies BCH codes depending on the data-width. For narrow-width values, Stealth ECC adopts the 3-bit correctable BCH code only for 32-bit meaningful data by storing 12-bit additional parity in MSB side (i.e., zero part), replacing zeros. Furthermore, with bitwise interleaved data placement between x4 DRAM chips, Stealth ECC is robust to a single DRAM chip error for narrow-width values. For full-width values, Stealth ECC adopts the SECDED code, maintaining DRAM reliability comparable to the conventional SECDED code. Since the additional parity bits are stored in the zero parts of narrow-width values, it does not require any additional storage overhead. As a result, thanks to the error mitigation for narrow-width values, Stealth ECC reduces the probability of system failure (caused by DRAM errors) by 47.9%, on average, with negligible performance overhead (0.9%, on average) and without storage overhead, compared to the conventional SECDED code.

ACKNOWLEDGMENT

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2020R1A2C2003500 and No. 2020R1A6A3A13064398), Samsung Electronics, and College of Information, Korea University. Sung Woo Chung and Young-Ho Gong are the co-corresponding authors of this paper.

REFERENCES

[1] S. Beamer, K. Asanovic, and D. Patterson, “The GAP benchmark suite,” arXiv:1508.03619, 2015.
[2] C. Bienia, S. Kumar, J. P. Singh, and K. Li, “The PARSEC benchmark suite: Characterization and architectural implications,” in *PACT*, 2008.
[3] N. Binkert et al., “The gem5 simulator,” *ACM SIGARCH Computer Architecture News*, vol. 39, no. 2, pp. 1-8, 2011.
[4] J. Bucek, K.-D. Lange, and J. Kistowski, “SPEC CPU2017: Next-generation compute benchmark,” in *ICPE*, 2018.
[5] J. Chen et al., “CARE: Coordinated augmentation for elastic resilience on DRAM errors in data centers,” in *HPCA*, 2021.
[6] S.-L. Gong, M. Rhu, J. Kim, J. Chung, and M. Erez, “Clean-ECC: High reliability ecc for adaptive granularity memory system,” in *MICRO*, 2015.
[7] J. Hong, H. Kim, and S. Kim, “EAR: ECC-aided refresh reduction through 2-D zero compression,” in *PACT*, 2018.
[8] X. Jian and R. Kumar, “Adaptive reliability chipkill correct (arcc),” in *HPCA*, 2013.
[9] J. Kim, M. Sullivan, and M. Erez, “Bamboo ECC: Strong, safe, and flexible codes for reliable computer memory,” in *HPCA*, 2015.
[10] J. Leskovec and R. Sosic, “SNAP: A general-purpose network analysis and graph-mining library,” *ACM Transactions on Intelligent Systems and Technology*, vol. 8, no. 1, pp. 1-20, 2016.
[11] V. Melikyan et al., “14nm educational design kit: Capabilities deployment and future,” *Small Systems Simulation Symposium*, 2018.
[12] P. J. Nair, D. H. Kim, and M. K. Qureshi, “ArchiShield: Architectural framework for assisting DRAM scaling by tolerating high error rates,” in *ISCA*, 2013.
[13] P. J. Nair, D. A. Roberts, and M. K. Qureshi, “Faultsim: A fast, configurable memory-reliability simulator for conventional and 3D-stacked systems,” *ACM Transactions on Architecture and Optimization*, vol. 12, no. 4, pp. 1-24, 2016.
[14] D. J. Palframan, N. S. Kim, and M. H. Lipasti, “COP: To compress and protect main memory,” in *ISCA*, 2015.
[15] N. Rohbani et al., “NVDL-cache: Narrow-width value aware variable delay low-power data cache,” in *ICCD*, 2019.
[16] D. Strukov, “The area and latency tradeoffs of binary bit-parallel BCH decoders for prospective nanoelectronic memories,” *Fourtieth Asilomar Conference on Signals, Systems and Computers*, 2006.
[17] 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 *ISCA*, 2012.
[18] Advanced Micro Devices, “BIOS and kernel developer’s guide (BKDG) for AMD family 15h models 00h-0fh processors,” 2013.
[19] Intel, “Intel® Xeon® Gold 6338 Processor,” 2021. [Available]: https://ark.intel.com/content/www/us/en/ark/products/212285/intel-xeon-gold-6338-processor-48m-cache-2-00-ghz.html