SAND2011-9099C

System Implications of Memory Reliability in Exascale Computing

Sheng Li[‡], Ke Chen[†‡], Ming-Yu Hsieh[¶], Naveen Muralimanohar [‡]
Chad D. Kersey[‖], Jay B. Brockman[†], Arun F. Rodrigues[¶], Norman P. Jouppi[‡]
[‡]Hewlett-Packard Labs, [†]University of Notre Dame,
[¶]Sandia National Labs, [‖]Georgia Institute of Technology
[‡]{sheng.li4, kec, Naveen.Muralimanohar, norm.jouppi}@hp.com
[†]{kchen2, jbb}@nd.edu, [¶]{myhsieh, afrodri}@sandia.gov [‖]cdkersey@gatech.edu

ABSTRACT

Resiliency will be one of the toughest challenges in future exascale systems. Memory errors contribute more than 40% of the total hardware-related failures and are projected to increase in future exascale systems. The use of error correction codes (ECC) and checkpointing are two effective approaches to fault tolerance. While there are numerous studies on ECC or checkpointing in isolation, this is the first paper to investigate the combined effect of both on overall system performance and power. Specifically, we study the impact of various ECC schemes (SECDED, BCH, and chipkill) in conjunction with checkpointing on future exascale systems. Our simulation results show that while chipkill is 13% better for computation-intensive applications, BCH has a 28% advantage in system energy-delay product (EDP) for memory-intensive applications. We also propose to use BCH in tagged memory systems with commodity DRAMs where chipkill is impractical. Our proposed architecture achieves 2.3× better system EDP than state-of-the-art tagged memory systems.

Categories and Subject Descriptors

C.0 [**Computer Systems Organizations**]: GENERAL

General Terms

Design, Performance

Keywords

Exascale computing, memory system, reliability, DRAM, ECC, chipkill, BCH, checkpointing, tagged memory

This material is based upon work supported by the Department of Energy under Award Number DE - SC0005026. The disclaimer can be found at http://www.hpl.hp.com/DoE-Disclaimer.html

Ke Chen and Jay Brockman are partially supported by the C2S2 Focus Center, one of six research centers funded under the Focus Center Research Program (FCRP), a Semiconductor Research Corporation entity.

1. INTRODUCTION

Among the obstacles on the road to exascale computing, system resiliency will be one of the toughest. In all hardware components, main memory is one of the most vulnerable components in the system. In existing terascale systems, hardware errors account for up to 60% of the total failures [5]. Of this, 40% of the hardware failures are memory related [30]. Its contribution will further increase in future systems not only because of the explosive increase in memory capacity for future exascale systems [5, 40]), but also because of the adoption of new technologies such as 3D stacking [36], greater device density, and lower supply voltage [39].

Error correction codes (ECCs) and checkpointing are two effective ways to protect a system from memory induced failures. By detecting and correcting memory errors on the fly, ECCs can significantly increase the reliability of the memory subsystem and thus increase the overall system resiliency. While ECCs protect systems from memory errors before they cause system failures, checkpointing recovers systems from failures not averted by the ECCs. Although warehouse data centers may allow individual nodes to be off-line when machines fail, exascale supercomputing leverages checkpointing to handle system failures since nodes are usually working together to solve large scale problems. In the presence of a failure, computation will restart from the last checkpoint, which wastes previous work on all the unaffected nodes.

While using different methodologies to maintain system resiliency, ECCs and checkpointing are tightly interrelated. Different ECC mechanisms can provide different memory reliability levels, meanwhile incurring different performance and power overhead. On the other hand, their reliability levels determine the checkpointing and rollback frequencies that in turn affect system's overall performance and power. For exascale computing systems running over a long time period (typically weeks or months), frequent checkpointing and rollback cause significant performance and energy overhead [15]. Thus, the increasing trend of memory error rates will affect exascale computing much more profoundly than it does in current petascale systems.

Memory reliability is even trickier for systems with tagged memory. Tagged memory adds an extension bit or bits to each memory word to describe its state. Systems with tagged memory have been proven especially effective for graph-oriented problems [4, 6] that involve intensive communication and synchronization between data items as well as irregular thread and memory behaviors. Tagged memory subsystems have been used in several important high performance computing systems, including HEP [42], Tera [10], lightweight chip multithreading (LCMT) [26–28], and more recently, the XMT [16]. However, supporting extension bits in commodity DRAM requires compromising the strength of ECC [16]. Continuing to support tagged memories in exascale systems without severely impacting performance or power demands exploring new coding scheme.

Since future exascale systems impose very tight power budgets while requiring a 1000-fold improvement in performance, understanding system implications of memory reliability and choosing an appropriate memory ECC mechanism for future exascale systems is critical. It requires a detailed analysis on overall trade-offs of performance and energy at the system level. In this paper we holistically assess the system implications of memory reliability for exascale systems with and without tagged memory. Our key contributions and findings are:

- We comprehensively study overall performance, energy, and efficiency implications of exascale systems, considering both pre-protection of memory systems using ECC and post-protection using checkpointing. We perform in-depth investigations not only on trade-offs between the achievable resiliency and the overhead of different ECC mechanisms but also the implications of checkpointing on overall system performance and energy when the ECC fails to prevent memory from failures.

- We identify the transition points where ECCs need to be upgraded to achieve better system resiliency on the road to exascale computing. Our results show that the current standard Simple Single Error Correction Double Error Detection (SECDED) will be insufficient for future exascale systems.

- We propose to use a Bose Chaudhuri-Hocquenghem (BCH) code as an alternative ECC mechanism for exascale systems to alleviate the high overhead of chipkill and to enable the use of commodity DIMMs in tagged memory systems without losing memory system reliability. Simulations show that for conventional non-tagged memory systems while chipkill is better for computation intensive applications, BCH can achieve 28% improvement of system energy-delay-product for memory intensive applications. For systems with tagged main memory our BCH proposal can achieve more than a 2.3× improvement on overall system EDP, compared to the current industry-standard implementation.

The remainder of this paper is organized as follows. After reviewing related work in Section 2, we describe the system implications of memory reliability in Section 3. In Section 4 we introduce BCH supported tagged memory. Section 5 describes the modeling framework we designed. In Section 6, we evaluate the performance, energy and EDP of exascale system with different ECCs. We conclude in Section 7 after discussing the evaluation results.

2. RELATED WORK

Main memory ECC has been intensively studied. IBM first proposed the concept of chipkill [13], and AMD invented an implementation [3] with the same chipkill-level protection while reducing required memory ranks by half. Ahn et al. [2] studied memory reliability under the context of bank-subsetting. Udipi et al. [45] proposed ECC and chipkill implementations for disruptive but efficient future memory devices. More recently, Yoon et al. [48] looked at using BCH ECC to protect non-volatile memory from wear-out errors. Wilkerson et al. [46] investigated using BCH ECC to protect the last level cache of processors to reduce the refresh frequency and thus to save power.

There also have been many proposals [11,15,49] on system checkpointing, focusing on improving coverage and reducing overhead in massively parallel processing (MPP) systems. Daly [11] proposed an analytical model to determine the best checkpointing interval, while [15] extended the model to further analyzed checkpointing overhead and proposed a two-level checkpoint scheme with non-volatile memory.

The Cray XMT machine [16] was the first to support tagged main memory while leveraging commodity DRAM DIMMs. However, their approach of increasing SECDED protection granularity from 64 bits to 128 bits imposes a significant system reliability loss for exascale systems.

Previous work looked at either system checkpointing or ECCs in isolation to protect systems from failures induced by memory errors. Our work is the first to consider ECCs and checkpointing in combination. We believe our work is also the first to consider BCH coding for systems with and without tagged main memory subsystems.

3. SYSTEM IMPLICATIONS OF MEMORY RELIABILITY

As the total memory capacity and the number of memory devices in a cluster increase, it is critical to keep the mean time to failure (MTTF) of individual devices high to avoid frequent failures. Unfortunately, soft errors in DRAM are expected to continue to increase with shrinking feature sizes [8]. Also, without significant improvements in fabrication processes or changes to DRAM organization, hard errors will be difficult to overcome. To the extent that the DRAM market is severely constrained by the cost per bit, major changes to memory are less likely, and hard errors due to failed row/column or stuck-at faults will continue to be an issue in future memories. Moreover, techniques such as 3D stacking, which will likely be adopted for DRAM [36] to meet the capacity demand, employ more silicon real estate and increase the probability of errors.

An effective way to alleviate the device reliability problem is improving the error detection and recovery in memory systems. In this section, we explore various root causes of failures in memory and evaluate the techniques to recover from them.

3.1 Memory Errors

Errors in memory manifest in different ways, from single bit errors to the failure of an entire DRAM chip. Table 1 lists the memory failure rate in FIT/Mbit (FIT = failures per billion hours) collected from major DRAM memory vendors for different technologies from 130 nm to 90 nm [9,41]. The

Error Type	Error Rate (FIT/Mbit)
Single-bit soft error	1000-5000 [44]
Logic soft chip error	$0.5e^{-3}$ - $5e^{-3}$ [9]
Chip hard error	13.7 [41]
Single-cell hard error	12.6 [41]
Row hard error	6.3 [41]
Column hard error	4.1 [41]
Row-Column hard error	4.2 [29]

Table 1: DRAM error mechanisms and error rates

memory cell soft error rate (SER) is in the range of $1e^3 - 5e^3$ FIT/Mbit [44], while the soft error rate in peripheral logic is an order of magnitude lower, and varies between $0.5e^{-3} - 5e^{-3}$ FIT/Mbit [9,12]. The occurrences of soft errors follow the exponential probability distribution $1 - e^{-\lambda t}$, where λ^{-1} is the mean time between failures. λ is tightly coupled to the transistor density. A single event induced by a cosmic ray or alpha article can cause multiple bit flips as the feature size goes down, although multiple errors may not present in the same memory word because of the interleaving of data bits in modern DRAM designs.

Hard error rate (HER) is two orders of magnitude lower than soft error rate (SER), reported to be 37 FIT/Mbit [41]. However unlike soft errors, they persist in memory and cause recurring errors. Depending upon the number of cells affected, more robust error correcting codes are required to tolerate hard errors. An extreme example of hard errors is a complete chip failure, also referred to as chipkill. The rate of chipkill errors was reported to be 13.7 FIT/Mbit [41].

As the failure rate values listed in Table 1 were reported in 2004, we scale them for future process technology to find the reliability requirements. Borkar et al. [8] suggests that the soft error rate doubles every technology generation. The lower signal-to-noise ratio for scaled technology will increase the vulnerability of stored data. Further, as density increases, a single alpha or neutron strike can cause multiple upsets. Thus, as technology advances from 45nm (petascale) to 11nm (exascale), the soft error rate may increase by 16×. On the other hand, the hard error rate in FIT/Mbit remains almost constant over time as claimed in [19]. Thus, we conservatively assume that with technology scaling, the total hard error rate is only proportional to the chip memory capacity and device counts, regardless of technology.

The net MTTF of a system due to memory related failures is proportional to memory capacity and node counts, and is given by Equation 1.

$$\begin{aligned} MTTF = 10^9/(&FailureRate* \\ &MemoryCapacityperNode * NumberofNodes) \end{aligned} \quad (1)$$

External factors such as memory utilization (i.e. number of memory accesses) and aging are also major contributors to memory error rates of DRAM [38]. For ease of measurement, memory utilization can be decomposed into indirect indicators such as CPU utilization and memory allocation. As a result, for each of the error mechanisms listed in Table 1, the error rate is actually a dynamic function of the three variables in Equation 2, where P_{error0} is the original error rate, U stands for CPU utilization, M stands for the memory allocation, and A is age. CPU utilization is the percentage of used CPU cycles out of available CPU cycles measured during runtime. Memory allocation is the total amount of memory the OS accesses for a workload. In our

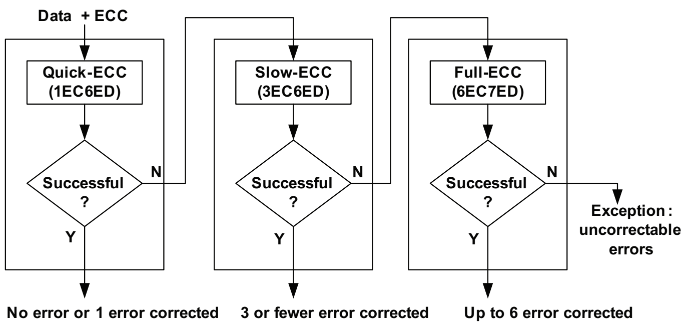


Figure 1: Flowchart of Staged BCH ECC in the memory controller

analysis, we collect the value for these two variables over our benchmark workloads, and assign system nodes with different workloads. Also, we assume the DRAM DIMMs in the system have a uniform aging distribution with the mean value of 3 years, and mark system nodes with specific ages.

$$P_{error} = F(U, M, A) = P_{error0} \times f_1(U) \times f_2(M) \times f_3(A) \quad (2)$$

$$\begin{aligned} f_1(U) &= 0.991 \times log_{10}(U/U_0) + 1 \\ f_2(M) &= 1.066 \times log_{10}(M/M_0) + 1 \\ f_3(A) &= 2.489 \times log_{10}(A/A_0) + 1 \end{aligned} \quad (3)$$

The dynamic function of error rates has a logarithmic dependence on U, M, and A as extracted from the dataset collected in [38], and is shown in Equation 3. Here U_0, M_0 are average CPU utilization and memory allocation across workloads, and A_0 is the average DRAM DIMM lifetime from DRAM vendors. Applying the three variable values in Equation 3, we get the factors for the DRAM failure rates, which are then fed into Equation 2 to calculate the error rate of each system node.

3.2 Protecting Memory from Failures

Hardware ECCs are an effective way to protect memory from failures. SECDED has already become standard in all modern servers. High end servers employ more robust chipkill-ECC memories, which can detect two and recover from one memory chip failure in a DIMM. In addition, double chipkill or Double Device Data Correction (DDDC) may be explored for future servers.

While error tolerance addresses the reliability challenge, chipkill or DDDC incurs high power and memory bandwidth overhead. Unfortunately, memory power and bandwidth are already a bottleneck in exascale machines [5]. Chipkill also places restrictions on DIMM configurations that can be used for servers. A straightforward way to implement chipkill is through *bit steering*: interleaving SECDED ECC words across multiple ranks [13] so that a failed DRAM chip will distribute its error bits into different ECC words. In this case, four SECDED codes of 72 bits interleaved across 288-bits are required to recover from a complete failure of a ×4 DRAM chip. While conceptually simple, this solution requires fetching pages from four ranks for every cacheline access. With diminishing locality in access stream, such overfetching is wasteful and dramatically increases DRAM energy [45]. A more sophisticated chipkill implementation employs symbol error correcting codes, as found in AMD Opteron processors [3]. Despite being a significant improvement, this approach still requires activating two ×4 ranks for each memory operation. This not only increases energy per access, it also hurts processor performance. This overhead will be even higher with future DRAM technologies that utilize a longer burst length.

3.3 Using BCH ECC as an Alternative

While replacing ECC with chipkill or DDDC provides high error coverage, chipkill is not mandatory in high performance clusters. Most modern supercomputers already have checkpointing and rollback mechanisms in place that can recover from system failures. Since frequent rollbacks are expensive, it is not desirable to leave the memory weakly protected. At the same time, memory protection codes should be fast and efficient for the majority of fault free accesses. Hence, by striking the right balance between memory protection and checkpointing, we can improve the overall energy efficiency of large systems.

Instead of updating ECC from SECDED to chipkill, we explore BCH code as an alternative. BCH codes are linear block codes defined over a finite Galois Field GF(2^m) with a generator polynomial, where 2^m represents the maximum number of code word bits. To correct t-bit errors in k-bit input data, BCH code typically requires $r = t * ceil(log_2 k) + 1$ check bits [34]. Therefore, it is not practical to implement BCH ECC for every 64 bit word. Fortunately, the size of an ECC code relative to that of the data word diminishes as the size of the data word grows. We thus choose the ECC word size as the size of the last level cache block, which is 64B for most current processors. Current DIMMs provide $1/8^{th}$ of the capacity for storing ECC check bits, thus a 64B memory block already has 64 bits reserved for ECC. A six error correction, seven error detection (6EC7ED) BCH requires 61 bits to protect a 64B memory block. Thus 6EC7ED BCH code can achieve high reliability similar to chipkill ECC. However, the main design issue with a strong BCH code is its complex error recovery procedure. To address this, we leverage the error detecting/correcting ideas presented in Free-p [48] and Hi-ECC [46] and implement three ECC logic paths to provide low ECC latency and high error coverage as shown in Figure 1: quick-ECC, slow-ECC and full-ECC. Quick-ECC processes the 6EC7ED BCH, detects up to seven errors, but corrects only one error without extra latency penalty. If quick-ECC identifies more than one error, it invokes slow ECC for correction. Slow-ECC can correct up to three bit errors. If slow-ECC detects more than three bit errors, it invokes full-ECC, which can correct up to 6 bit errors. Section 5 provides detail analysis on area, latency, and power for SECDED, chipkill, and BCH ECC.

3.4 Effect of Memory Failures on System Performance

Main memory errors contribute to 25% of total system failures [5, 30, 37]. Different ECC codes can provide various protection abilities for a system and result in different system failure rates, or MTTF. To recover from failures not corrected by ECC, a checkpoint/rollback scheme is used. The stronger the ECC, the fewer the failures, hence leading to less rollbacks. On the other hand, ECC hardware itself incurs overhead even in fault free operations, especially for chipkill.

We evaluate the impact of trade-offs between the achievable resiliency and the overhead of three ECC mechanisms (SECDED, BCH, and chipkill) on the performance and power of future exascale systems. We model state-of-the-art highly scalable two level checkpointing [15], where periodic synchronous snapshots are made in both local nodes and neighboring nodes. Unless a node has failed completely, local rollbacks are sufficient to recover from failures. In rare cases where network or power failures prevent local rollbacks, global checkpoints can be used to start a new node and resume execution. T_{total}, the execution time of such a system, consists of four parts: the original computation time for a workload, the time spent on generating checkpoints, the time spent on recovering from a system failure, and finally the extra cost for a global checkpoint failure.

T_{total}	Total execution time including all the overhead
T_S	Native execution time of a workload
p_L	Percentage of local checkpoints
p_G	1 - p_L , the percentage of global checkpoints
τ	Interval between two local checkpoints
δ_L	Local checkpoint overhead (dumping time)
δ_G	Global checkpoint overhead (dumping time)
δ_{eq}	Equivalent checkpoint overhead in general
R_L	Local checkpoint recovery time
R_G	Global checkpoint recovery time
R_{eq}	Equivalent checkpoint recovery time in general
q_L	Percentage of failure covered by local checkpoints
q_G	1 - q_L ,the percentage of failure that has to be covered by global checkpoints
$MTTF$	System mean time to failure
P_S	Native execution power of a workload
P_{ckpt}	Power during checkpointing in general
P_R	Power during recovery
$P_{ckpt,L}$	Local checkpoint power overhead

Table 2: Timing and power related parameters for two-level (local/global) checkpointing scheme

Thus the total execution time comes to:

$$T_{total} = T_S + \frac{T_S}{\tau}(\delta_{eq}) + (\frac{1}{2}(\tau + \delta_{eq}) + R_{eq})\frac{T_{total}}{MTTF} + \frac{P_L q_G}{2p_G}(\tau + \delta_L)\frac{T_{total}}{MTTF} \quad (4)$$

where the parameters are detailed in Table 2.

Similar to the timing model, we derive a energy model based on the bandwidth and the power of all hardware components, as well as the checkpointing/recovery distribution over two levels. The total execution energy consumption is

$$\begin{aligned} E_{total} &= P_S T_S + (P_{ckpt}\delta_{eq})\frac{T_S}{\tau} \\ &+ (\frac{1}{2}(P_S\tau + P_{ckpt}\delta_{eq}) + P_R R_{eq})\frac{T_{total}}{MTTF} \\ &+ \frac{P_L q_G}{2p_G}(P_S\tau + P_{ckpt,L}\delta_L)\frac{T_{total}}{MTTF} \end{aligned} \quad (5)$$

4. RELIABILITY OF EXASCALE SYSTEMS WITH TAGGED MAIN MEMORY

When making changes to ECC codes in DRAM, it is worth considering their effect on tagged main memory - a requirement that caters to graph-oriented applications. These applications can achieve significant improvements in performance (1.5× in average) [26–28] when executed on systems with tagged memory. In a 64-bit tagged memory sub-system, the fundamental storage unit is called *extended double word* or `Xdword`. It consists of a 64-bit double word (`dword`) together with an extra bit called *extension bit (Xbit)*. By using the extension bit in tandem with mode fields within the `dword`, a set of memory states for the `dword` can be defined. This supports synchronization at the memory word level by allowing memory operations to execute conditionally, depending upon the state of the memory that they are attempting to access. When a memory operation cannot execute because the location being accessed is not in the correct state, the thread responsible for that operation gets blocked. Once the memory state is restored, the thread will continue its execution.

It is relatively easy to implement tagged memory support in mainstream processors [27], however, the extra bit required for each word in DRAM requires custom DRAM dies. One way to address this problem is by leveraging a portion of the storage for ECC to store the extension bit. The CRAY XMT supports tagged main memory in commodity DIMMs through this technique. The XMT machine enlarges the ECC protection granularity from 64 bits to 128 bits, resulting in savings of three bits for every 64-bit memory word. The memory controller is then modified to treat the saved bits in the ECC chip as extension bits. Unfortunately, increasing the ECC word from 64 bits to 128 bits results in 2.25× decrease in the memory system reliability based on our Monte Carlo simulation. With resiliency being a first order design constraint for exascale systems, providing error coverage lower than traditional ECC is not attractive for future systems.

To solve this problem, we propose to use BCH as the ECC on the systems with tagged memory. We choose the ECC word size as the size of last level cache block, i.e. 64B for most current processors. A 5EC6ED BCH code requires only 51 ECC bits to protect every 64 byte data block and the associated 8 extension bits. Therefore, 13 bits in the ECC section can be saved for each cache line sized data by using a BCH code. Error detection and correction is performed on both the data bits and extension bits. We compare the system resiliency, performance, and energy of the XMT implementation and our proposal in Section 6.

5. MODELING METHODOLOGY

To evaluate the effect of different levels of memory protection on the performance and power of an exascale system, we use a modeling framework shown in Figure 2. The framework consists of four major parts: 1) a Monte Carlo simulator to study the interplay between different types of memory errors; 2) an extended McPAT framework for studying the energy, latency, and area of future manycore processors with different ECC techniques; 3) performance simulators to evaluate both conventional non-tagged memory systems and tagged memory systems; and 4) a performance and power analyzer for exascale systems with different ECC mechanisms.

The Monte Carlo simulation is used to generate system failure rates under different error types and ECC techniques. This is important since multiple types of errors can occur at the same time and different ECC types react differently to these errors. The simulator injects different types of errors into the memory based on the failure rates given in Table 1. The error injection follows an exponential distribution. Also,

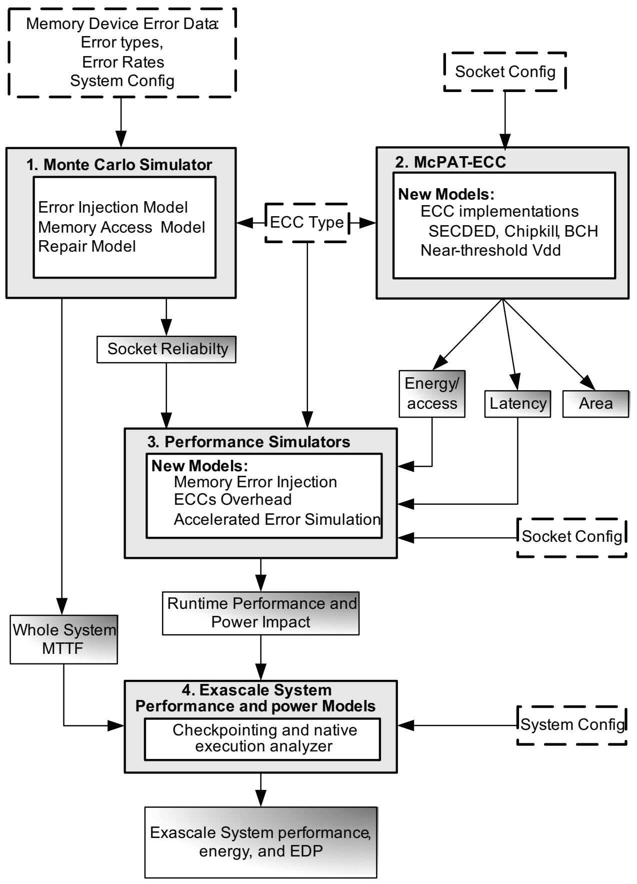


Figure 2: Simulation framework for evaluation of system performance, energy, and EDP from memory errors and ECC types.

the scaling trends in Section 3 combined with external factor dynamic functions as in Equations 3 and 2 are used to obtain the error rates with given technology nodes, workloads, and memory system configurations. We then scan through DRAM chips across a DIMM, and collect the errors in each memory word. We also differentiate recurring hard errors and intermittent soft errors when calculating total errors per cacheline access in burst transfers. The simulator knows exactly the number and types of the errors injected during simulation and the error detection and correction abilities of a particular ECC scheme. If the total error bits exceed the detection capability (silent error) or correction capability (uncorrectable error) of the ECC scheme, then it is considered as an error to cause a system failure. Finally, the failure rates (or MTTFs) of different ECC types are obtained from the number of uncorrectable failures out of the total memory accesses during the average lifetime of modern DIMMs (3 years) [38] with a memory utilization percentage of 31% [15].

McPAT is an integrated power, area, and timing modeling framework for multicore and manycore processors [23, 24]. We extend McPAT to model power, area, and timing of future manycore processors equipped with different ECCs (SECDED, chipkill, and BCH) for exascale computing. When modeling ECC hardware in memory controllers, we decompose it into gate level circuit blocks and use McPAT's circuit models estimate area, latency, and energy per access of each type of ECC hardware. Of the various ECC

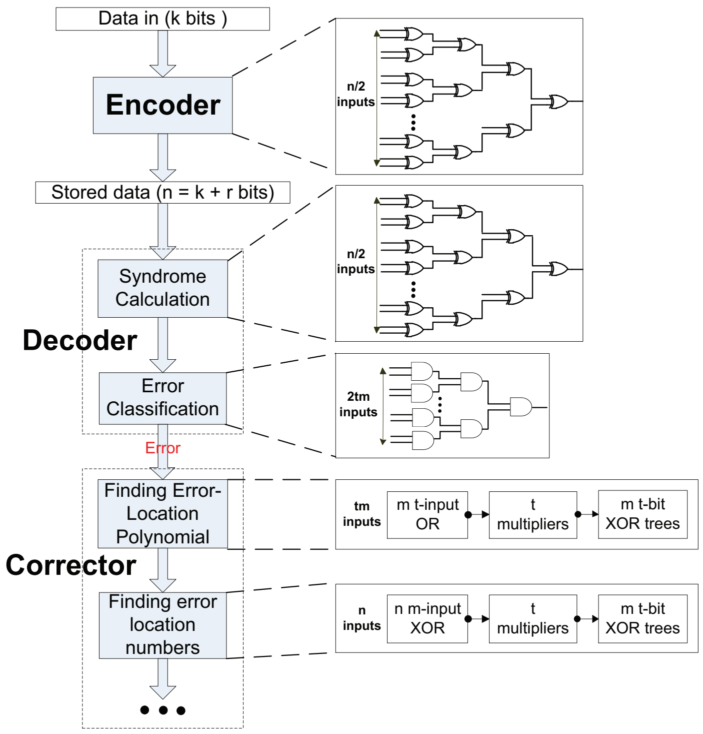


Figure 3: BCH ECC circuit with gate level breakdown

schemes modeled, BCH is the most complex code and requires more gates. To evaluate the feasibility of using BCH in memory controllers, we accurately evaluate the overhead of the encoder, decoder, and corrector logic for BCH. The decoding process itself has three steps: 1. syndrome calculation (and error classification), 2. calculating the coefficients of the error-location polynomial, and 3. finding error locations by solving the roots of the polynomial. We model a pipelined BCH architecture [43] and size the devices to reduce latency.

Figure 3 shows the gate level model for various stages in BCH hardware, where we assume t bit correction for n bit information code, and $m = log2(n)$. According to [43], the total number of logic gates in a BCH code decoder and correction logic is dominated by XOR gates in the error correction step, and thus scales with mt^2n. The area overhead of these stages can be calculated using Equation 6, where F is the process feature size.

$$A_{BCH} = (A_{xor} \times N_{xor} + A_{and} \times N_{and} + A_{or} \times N_{or})) \times F^2 \quad (6)$$

The latency of the decoder is dominated by the *finding error-location* step and scales with tm. This is because the t iterations cannot be fully parallelized. The total latency is derived by finding the latency of the critical path. The energy per BCH operation is calculated based on the number of active gates in each step. Table 5 shows the power and timing results for SECDED and BCH, with a detailed explanation in Section 6 .

Once we find the error rate and the overhead for correcting them, the next step is to analyze their effect on a large system. Since simulating an exascale system is not feasible, we first simulate the performance and power of an individual node/socket with different ECC schemes. For the systems with conventional non-tagged memory subsystems, we use McSim [1], a Pin [31] based simulator. McSim is event driven and cycle accurate. It models in-order cores, caches, directories, on-chip networks, memory controllers, and DRAM banks. To model systems with tagged main memory, we use the Structural Simulation Toolkit (SST) [18, 35]. SST is an open, modular, and parallel simulation framework. It includes a number of processor, memory, and network components. For this study, we develop a chip multiprocessor (CMP) component linked with the DRAMSim2 [32] component to model a core cluster (Figure 4) connected to a memory system. The CMP component can run benchmarks using full/empty bits by translating atomic memory operations into equivalent full/empty bit operations. The extended McPAT framework provides power, area, and timing information of the simulated processors for both simulators. The error injection model (as in the Monte Carlo simulation) is integrated into both simulators to enable our research on ECC hardware runtime overhead. The performance and power values of individual sockets along with the MTTF of memory calculated through Monte Carlo simulation are then used in the performance and power analyzer as shown in Figure 2 along with Equation 4 and 5 to find the exascale system performance.

	1petaFLOPs	1exaFLOPs
FLOPS	10^{15}	10^{18}
Year	2008	2018
Technology node	45 nm	11 nm
Number of nodes	20, 000	100, 000
Memory per node	4GB	640GB
Memory bandwidth/node	10 GB/s	4 TB/s
Node performance	50 GF	10 TF
Network bandwidth/node	3.5GB/s	4TB/s
Memory induced MTTF	1.3 days	1.1 hour

Table 3: Specifications of the projected exascale system and a current petascale system.

6. EVALUATION

Table 3 shows the specifications for a current petaflop system and a projected exascale system. The projections and scaling assumptions are based on ITRS [39] and (more aggressive) exascale computing studies [5, 8, 40]. An exascale system will likely have 100,000 nodes, with each node providing 10 teraFLOPS of processing capability. The memory capacity for each node will be 640GB, a 160× increase compared to current petaFLOP systems. Memory bandwidth requirement will be 4 TB/s, a 400× increase from current systems. The inter-node network bandwidth is projected to be in a range of 200GB/s [15] to 4TB/s [40] based on whether an optical network will be aggressively used. Using the Monte Carlo simulation discussed in Section 5, we compute the memory failure rate to be around 400 per billion hours per GB for a system that supports 64-bit SECDED ECC. This is consistent with the data measured in actual systems [29, 30]. Thus, taking into account the total number of computation nodes, the memory induced MTTF for a petascale system is 31 hours (1.3 days). For an exascale system, the memory induced MTTF is around 1.1 hour [1]. Both MTTFs are well aligned with the expectations in [40].

[1]Petascale system uses SECDED, while we assume the exascale system uses chipkill since checkpointing overhead under SECDED is too high for exascale systems, as we show in Section 6.

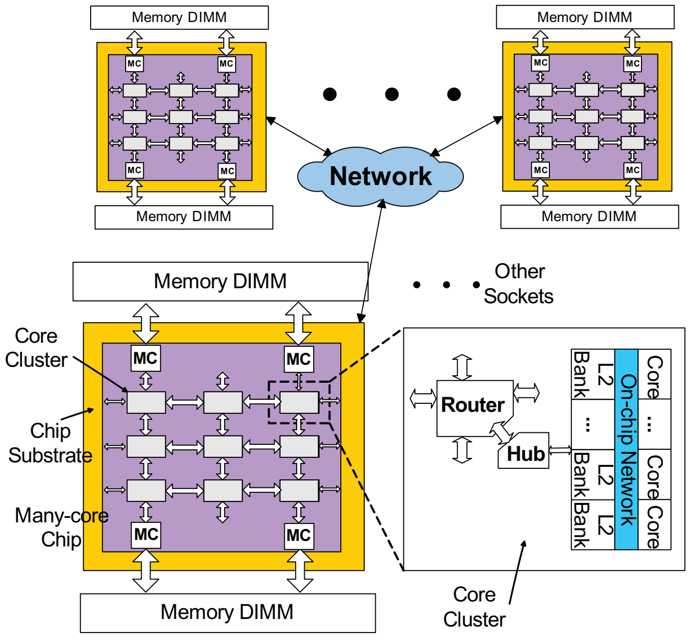


Figure 4: **Conceptual view of the exascale system architecture.**

Core	
Technology (nm)	11
L1 Cache	32KB, 16way, 64B block
L2 Cache	512KB, 16way, 64B block
Clock rate (GHz)	2
Peak Perf. w. SIMD (GFLOPs)	8
Power (near-threshold op) (W)	0.62
Area (mm^2)	1.5
Cluster	
Core count	32
Memory Controllers	3 dual-channel
Memory type	DDR4-4266
Cluster Power (W)	21
Total threads	256
Peak Perf. w. SIMD (GFLOPs)	256
FLOPs / Mem BW (ops/B)	2.5
Memory DIMMs	DRAM / PCRAM
Energy including I/O (pJ/bit)	7.1 / 9.2
Leakage Power/DIMM (mW)	129 / 36

Table 4: Parameters of the simulated core cluster.

Figure 4 shows the block diagram of an exascale system. It consists of multiple nodes connected using fast system interconnects. Each node consists of one or more manycore processors. We model a clustered manycore processor similar to Koka et al [21].

6.1 Experiment Setup and Modeling Results

In order to keep the simulation time reasonable, we first find the performance of an individual socket by simulating a cluster of cores (as shown in Figure 4). We choose a dual pipelined core similar to Niagara-2 and augment each core with a 4-way SIMD unit. To meet the performance target without breaking the power envelope, processors in exascale systems will likely be working at near-threshold supply voltage. We use McPAT [23,24] to estimate the power, area, and clock rate of these cores when operating at near-threshold supply voltages. Table 4 lists the architecture parameters and the results obtained from McPAT for processor cores at an 11nm technology. Since each core achieves 8 GFLOPs, a 10 TFLOPs node listed in Table 3 requires around 1280 cores. Assuming that each node has eight sockets, a socket will have 160 cores in a die (of size 240 mm^2). If a core cluster contains 32 cores, such a socket can have five clusters built on the chip substrate. According to the system specifications in Table 3, each core cluster needs 205 GB/s of memory bandwidth. Based on the JEDEC roadmap [20] and the clock rate of cores, DDR4-4266 equivalent memory can be used. Thus, the core cluster needs 3 dual-channel memory controllers with each memory channel containing a single rank DDR4-4266 DIMM. The memory controller is modeled to support a combination of open-page and close-page policies. A DRAM page gets closed during a precharge command unless there are more pending requests to the same pages in the memory controller queue. Chipkill is modeled according to the optimized design in AMD Opteron processors [3]. When chipkill is not used, cachelines are interleaved across all memory channels. However, when chipkill is used, fetching a single cacheline needs to invoke two memory channels to finish the chipkill operation. Hence, without enough locality, all the extra bits fetched into the row buffers in all the active DIMMs will go waste. Further, this approach blocks parallel accesses to various DIMMs, increasing the queuing delay. The modeled memory controller supports a DRAM power down mode during which DRAM chips consume 1/5th of normal static power and need two cycles to enter and exit the state [2].

As mentioned in Section 3, checkpointing can be used to recover the system state in the event of an uncorrectable error. We model an advanced two-level checkpointing scheme and store checkpoints both locally within a node and globally in neighboring nodes [15]. Both local and global checkpointing are assumed to be stored in emerging non-volatile PCRAM since hard disk drives (HDD) have been proved to be too slow for exascale checkpointing [15].

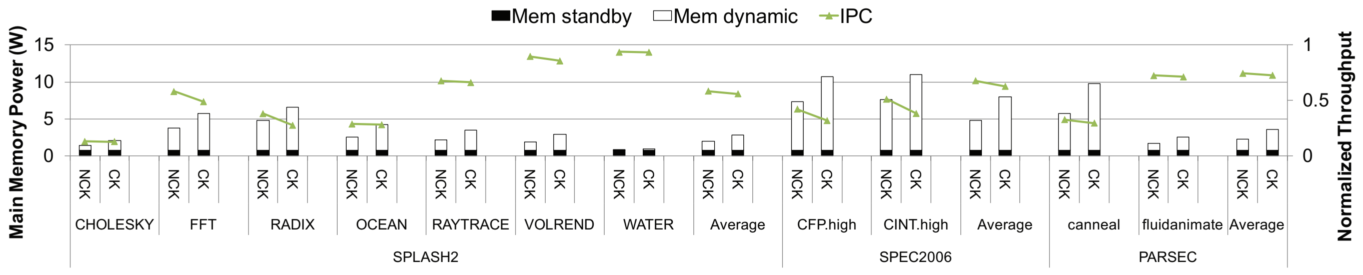
We run CACTI [25] and PCRAMSim [14] to compute the

SECDED	Encoder	Decoder	Corrector
Gate number	288/1088	552/1744	
EPA (pJ)	0.014/0.053	0.017/0.059	
Latency (ns)	0.055/0.065	0.065/0.075	0.020/0.023
BCH	Encoder	Decoder	Corrector
Gate number	25k/21k	26k/22k	201k/168k
EPA (pJ)	1.2/1.0	1.3/1.1	9.3/7.7
Latency (ns)	0.081/0.081	0.10/0.10	1.2/1.0
BCH corrector	t=6/5	t=3	t=1
EPA (pJ)	9.3/7.7	4.9	2.1
Latency (ns)	1.2/1.0	0.65	0.28
Cycles	7/6	4	2

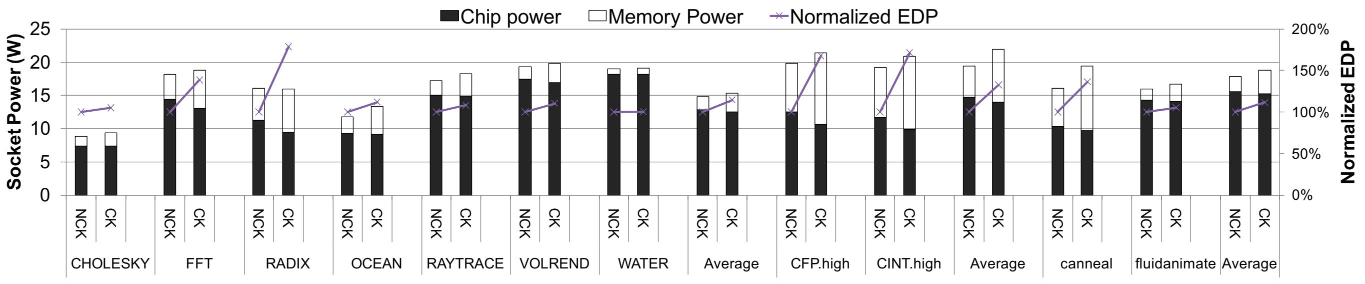
Table 5: Energy per access (EPA), gate count, and timing of the SECDED hardware (top), the BCH decoder (middle), and the staged BCH corrector (bottom) at 11nm. Values in left/right pairs are 64-bit/128-bit for SECDED and 6EC7ED/5EC6ED for BCH, respectively. The 128-bit SECDED and 5EC6ED are used for tagged memory only. *t* in staged BCH (bottom) denotes the correcting ability of BCH corrector. Cycles are for the 4GHz DDR4-4266 channel.

SPLASH-2				**SPEC CPU2006**	
Application	Dataset	Application	Dataset	Set	Applications
Barnes	16K particles	Cholesky	tk17.O	CINT	
FFT	1024K points	Radiosity	room	high	429.mcf, 462.libquantum, 471.omnetpp, 473.astar
FMM	16K particles	Raytrace	car	med	403.gcc, 445.gobmk, 464.h264ref, 483.xalancbmk
LU	512×512 matrix	Volrend	head	low	400.perlbench, 401.bzip2, 456.hmmer, 458.sjeng
Ocean	258×258 grids			CFP	
Radix	8M integers			high	433.milc, 450.soplex, 459.GemsFDTD, 470.lbm
Water-Sp	4K molecules			med	410.bwaves, 434.zeusmp, 437.leslie3d, 481.wrf
				low	436.cactusADM, 447.dealII, 454.calculix, 482.sphinx3

Table 6: SPLASH-2 datasets and SPEC CPU 2006 application mixes for high, med, and low memory bandwidth.



(a) Memory power and IPC (IPC higher is better) of the simulated 32-core cluster



(b) Power and EDP (lower is better for power and EDP) of the simulated 32-core cluster

Figure 5: Power and performance of applications on systems with and without chipkill-level reliability. In both figures CK means with chipkill, and NCK means without chipkill.

energy per access of the main memory DRAM DIMMs and checkpointing PCRAM DIMMs. Based on current high-speed link energy consumption (8 pJ/bit at 40nm) in [22] and scaling trends shown in [33], we conservatively assume that the I/O energy consumption of a memory channel is 2pJ/bit. The results of this analysis are listed in Table 4 and are used in our core-cluster simulations later in this section.

Our modified McPAT is used to compute the area, energy, and timing of the manycore processors with the overhead of different ECC schemes. As shown in Table 5, the BCH code incurs significant area and energy overhead compared to SECDED. However, the dynamic power at 11nm of the delay-optimized BCH for each channel is only 28mW, which is negligible when compared with the processor power. When there is no error, the SECDED and BCH have the same 1 cycle latency for the 4GHz memory channel. Although BCH takes 7 cycles of latency to perform the full 6EC7ED, this overhead is greatly reduced by applying staged BCH ECC as described in Section 3. As shown in Table 5 (bottom), the single bit correction incurs only 1 extra cycle of latency compared to SECDED.

SPLASH-2 [47], PARSEC [7], and SPEC CPU2006 [17] benchmark suites were used to evaluate the performance and power of a single node. We used all SPLASH-2 applications and five of the PARSEC applications (canneal, streamcluster, blackscholes, fluidanimate, and swaptions). We used the simlarge dataset for PARSEC applications, and the datasets used for SPLASH-2 applications are summarized in Table 6. We used the SPEC CPU2006 benchmark suite to measure the system performance on consolidated workloads. We picked 12 applications from both CINT (integer benchmarks) and CFP (floating point benchmarks), and made 3 groups each, 4 applications per group, based on their main-memory bandwidth demand [17] as shown in Table 6. To evaluate the performance and reliability of systems with tagged main memory, we used SPLASH-2, PARSEC, and the Multi-Threaded Graph Library (MTGL) [6] benchmark suite. MTGL is a group of graph-oriented applications designed to run on shared-memory platforms. We selected four applications from the MTGL: breadth-first search (BFS), connected component (CC), PageRank, and RMAT generation. Note that the RMAT generation was written from the ground up to use full/empty bits. We skipped the initialization phases of each workload and simulated 2 billion instructions or until the program finishes, whichever comes first.

6.2 Single Socket Results

Figure 5 presents the simulation results of the 32-core cluster in a socket as listed in Table 4. As shown in prior analysis, the latency and power overhead caused by pure ECC hardware is negligible even for BCH ECC. Our simulation results confirm that even with 20× accelerated error injection, BCH and SECDED exhibit very similar performance and energy consumption, with BCH causeing less than 1% degradations in performance and power compared to SECDED. Thus, we group the ECC techniques of 64b SECDED, 128b SECDED, and BCH as a non-chipkill category and compare it with chipkill. Figure 5 shows the comparisons between the non-chipkill system and the chipkill system, with both systems having non-tagged memory. For computation-intensive benchmarks such as WATER and fluidanimate, the difference between systems with chipkill and without chipkill is very small. This is because memory is seldom used, and therefore the high overhead of chipkill does not cause a significant change in the system performance and power. However, for memory-intensive benchmarks such as CFP.high and RADIX, chipkill incurs up to a 38% performance overhead and a 35% increase in memory power as shown in Figure 5(a). This is because for every cacheline needed, chipkill reads twice the data, wasting power and bandwidth. In particular, activating twice the number of DIMMs to access a single cacheline can block subsequent reads to different memory locations. This also wastes power since programs with low row buffer hit rates will end up wasting all excess cache lines fetched to the DRAM row buffers. As shown in Figure 5(b), for memory-intensive benchmarks, the system with chipkill demonstrates an EDP degradation of up to 60% (e.g. CFP.high), compared to systems without chipkill.

6.3 From Single Socket to Exascale System

The overall goal of this work is to find the reliability technique with optimal energy delay product, which is done in two steps. First, we investigate checkpointing overhead caused *purely* by reliability levels provided by different ECCs. If an ECC cannot provides enough reliability, the memory induced failures cause the system to spend an unacceptable portion of its time checkpointing and rolling back. Thus, we can find where, on the roadmap to exascale computing, the system needs to upgrade to a stronger ECC scheme to keep the checkpointing overhead tolerable. Once the ECC schemes that can be used for exascale computing are identified, the second step is to factor in ECC overhead on native fault-free execution so that the best ECC technique can be selected from the still available ones. When studying system-level performance, we scale the socket level simulations to the exascale system level for this analysis and consider the system-level communication overhead based on projections from Kogge et al [5]. We also factor in the parallel efficiency when scaling from a single socket to the entire exascale system.

Figure 6 shows the energy, performance, and energy-delay product of checkpointing overhead on systems with different scales and ECCs. The overhead grows superlinearly from 64 petaFLOPs to exaFLOPs, caused by rapid increase of the failure rate and the resulting higher frequency of checkpointing and recovery. Although 64-bit SECDED is still possible at exascale, it wastes lots of machine time and power since the EDP overhead for checkpointing is more than 130% of the native execution. The 128-bit SECDED for tagged memory is even worse for exascale computing because of its associated 347% checkpointing overhead on EDP over the native

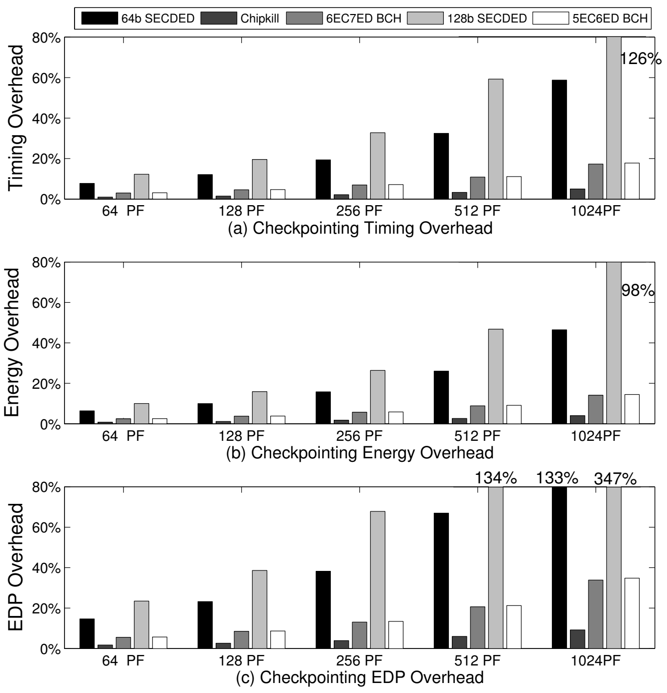
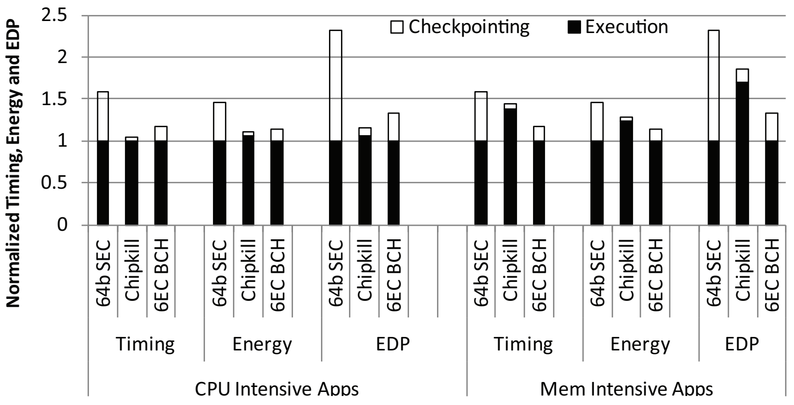


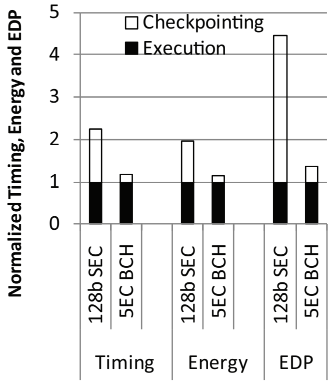
Figure 6: Checkpoint (and rollback) overhead of systems with different ECC schemes scaling from 64 petaFLOPs to 1 exaFLOPs for timing, energy, and EDP (lower is better). The two level non-volatile DIMM based checkpointing mechanism [15] is used. Checkpoint size is assumed to be 30% of total memory capacity as indicated in [15]. All numbers are normalized against that of native execution without checkpointing. Numbers greater than 100% indicate that the system spends more time/energy on checkpointing than on the native execution (the real work).

execution. As a result, chipkill and BCH are the good options for future exascale computing.

Figure 7 shows a complete view of overall system performance, energy and EDP at exascale, including both native execution and checkpointing operations. As shown in Figure 7(a), for both computation-intensive and memory-intensive workloads running on systems with conventional non-tagged main memory, chipkill and 6EC7ED BCH have significantly less total EDP than 64-bit SECDED. For computationally-intensive workloads, although chipkill requires more energy than BCH for native execution, its higher MTTF results in higher reliability and lower checkpointing overhead, which leads to a 13% lower EDP than that of BCH. However, for memory-intensive workloads, BCH is much better in power and performance of native execution, which leads to 28% advantage on EDP over chipkill even when the larger checkpointing overhead of BCH is considered. For systems with tagged main memory, because the system needs to pack the extension bits in the ECC section, traditional chipkill implementation is impractical, and only the 128-bit SECDED and 5EC6ED ECC schemes can be used. The 5EC6ED BCH gives 2.3× better EDP than the 128-bit SECDED does. Thus, with BCH ECC, the system with tagged main memory can leverage commodity DIMMs while maintaining acceptable system resiliency and checkpointing overhead.



(a) Systems with conventional non-tagged memory.



(b) Systems with tagged memory

Figure 7: Complete view of overall system timing, energy, EDP (lower is better) at exaFLOPs for systems with conventional non-tagged main memory and tagged memory. All numbers are normalized against native execution of SECDEDs (64 bit and 128 bit). SECDEDs are still included since it is possible for exascale computing although they are not practical because of the high checkpointing overhead. For systems with tagged memory, since chipkill is impractical, the difference between computation-intensive and memory-intensive workloads is negligible and is not shown.

Figure 7 shows that memory bandwidth utilization has a big impact on the overall system performance of chipkill and BCH, which is within expectations since benchmarks with high memory bandwidth demand and utilization place higher pressure on memory systems with chipkill than on systems without chipkill. Figure 8 gives the system time, energy and EDP ratio between chipkill and 6EC7ED BCH ECC as we sweep system-level effective memory bandwidth utilization. System-level effective memory utilization is actually the product of the memory utilization of a node or socket (simulated) and the system level parallel efficiency beyond a single node. Because of the huge design space of this study, Figure 8 is obtained by curve fitting our simulated results together with network/communication projections from [5]. An analytical model may be possible but is beyond the scope of this paper. Besides confirming that chipkill is better for cpu-intensive benchmarks while BCH is better for memory-intensive benchmarks, Figure 8 further reveals the crossover points at memory utilization of 59%, 27% and 55% for time, energy, and EDP, respectively.

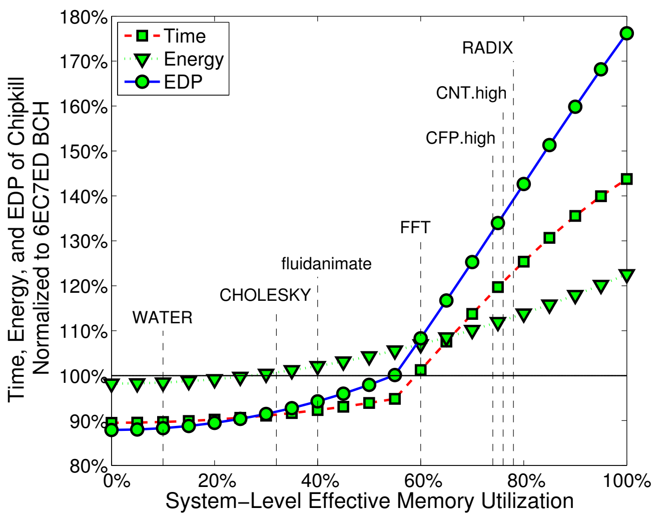


Figure 8: Normalized overall system performance, energy and EDP (lower is better) of chipkill over 6EC7ED BCH (including both native execution and checkpointing operation). The figure is based on curve-fitting the simulated data, with representative simulation data points. The portions of the curves with less than 10% or more than 80% of the system-level effective memory bandwidth utilization are backward and forward projections, respectively.

7. CONCLUSIONS

Both memory ECCs and checkpointing will be important tools providing reliability and availability in future exascale systems. However, while providing different levels of memory reliability, different memory ECCs incur different performance and power overhead. Moreover, they impact checkpointing behavior that in turn affect system performance and power. This paper is the first to assess the implications of memory reliability at exscale system level, considering both pre-protection (ECC) and post-protection (checkpointing) mechanisms rather than looking at either in isolation.

With this work, we have identified the key relationships between pre-protection (ECCs) and post-protection (checkpointing) to provide a reliable system, and proposed BCH as an alternative memory ECC to improve system level performance and energy efficiency at exascale computing where SECDED will incur significant overhead. The proposed BCH ECC also enables the smooth adoption of tagged memory in future exscale systems for solving important graph-oriented problems. Using our evaluation framework, we found that careful system level considerations are necessary to achieve the best system resiliency. Our results show that for systems with conventional non-tagged main memory subsystems, while chipkill is 13% better for computation-intensive applications, BCH can achieve a 28% improvement of system EDP for memory-intensive applications. For systems with tagged main memory, BCH is 2.3× better compared to the state-of-the-art 128-bit SECDED implementation while still leveraging commodity DIMMs.

8. REFERENCES

[1] "McSim: A Manycore Simulation Infrastructure," http://scale.snu.ac.kr/mcsim.

[2] J. H. Ahn, N. P. Jouppi, C. Kozyrakis, J. Leverich, and R. S. Schreiber, "Future Scaling of Processor-Memory Interfaces," in *Supercomputing Conference*, 2009.

[3] AMD, "BIOS and Kernel Developer's Guide for AMD NPT Family 0Fh Processors, Technical Report," Nov. 2009.

[4] D. A. Bader, G. Cong, and J. Feo, "On the architectural requirements for efficient execution of graph algorithms," in *ICPP '05: Proceedings of the 2005 International Conference on Parallel Processing*, 2005, pp. 547–556.

[5] K. Bergman, *et al.*, "ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems." 2008, DARPA IPTO sponsored report.

[6] J. Berry, B. Hendrickson, S. Kahan, and P. Konecny, "Software and Algorithms for Graph Queries on Multithreaded Architectures," in *2007 IEEE International Parallel and Distributed Processing Symposium*, 2007, p. 495.

[7] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The PARSEC Benchmark Suite: Characterization and Architectural Implications," in *PACT*, 2008.

[8] S. Borkar, "The Exascale Challenge," in *Asia Academic Forum*, Nov 2010.

[9] L. Borucki, G. Schindlbeck, and C. Slayman, "Comparison of Accelerated DRAM Soft Error Rates Measured at Component and System Level," in *Proceedings of 46th Annual International Reliability Physics Symposium*, 2008.

[10] Cray Corporation, "Cray MTA-2 System."

[11] J. T. Daly, "A Higher Order Estimate Of The Optimum Checkpoint Interval For Restart Dumps," *Future Gener. Comput. Syst.*, vol. 22, pp. 303–312, February 2006.

[12] T. J. Dell, "System RAS Implications of DRAM Soft Errors," *IBM Journal of Research and Development*, vol. 52, no. 3, pp. 307–314, 2008.

[13] T. Dell, "A White Paper On The Benefits Of Chipkill-Correct ECC for PC Server Main Memory," IBM Microelectronics Division," Technical Report, Nov. 1997.

[14] X. Dong, N. P. Jouppi, and Y. Xie, "PCRAMsim: System-Level Performance, Energy, and Area Modeling for Phase-Change RAM," in *Proceedings of the 2009 International Conference on Computer-Aided Design*, ser. ICCAD '09. New York, NY, USA: ACM, 2009, pp. 269–275. [Online]. Available: http://doi.acm.org/10.1145/1687399.1687449

[15] X. Dong, N. Muralimanohar, N. Jouppi, R. Kaufmann, and Y. Xie, "Leveraging 3D PCRAM Technologies to Reduce Checkpoint Overhead for Future Exascale Systems," in *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, 2009.

[16] J. Feo, D. Harper, S. Kahan, and P. Konecny, "ELDORADO," in *Proceedings of the 2nd conference on Computing frontiers*, Ischia, Italy, 2005, pp. 28–34.

[17] J. L. Henning, "Performance Counters and Development of SPEC CPU2006," *Computer Architecture News*, vol. 35, no. 1, 2007.

[18] M.-y. Hsieh, A. Rodrigues, R. Riesen, K. Thompson, and W. Song, "A Framework for Architecture-Level Power, Area, And Thermal Simulation and Its Application to Network-on-Chip Design Exploration," *SIGMETRICS Perform. Eval. Rev.*, vol. 38, pp. 63–68, March 2011. [Online]. Available: http://doi.acm.org/10.1145/1964218.1964229

[19] B. Jacob, S. Ng, and D. Wang, *Memory Systems: Cache, DRAM, Disk.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007.

[20] JEDEC, "http://www.jedec.org/."

[21] P. Koka, *et al.*, "Silicon-Photonic Network Architectures For Scalable, Power-Efficient Multi-Chip Systems," *ISCA 2010*, vol. 38, pp. 117–128, June 2010.

[22] H. Lee, *et al.*, "A 16Gb/s/link, 64GB/s Bidirectional Asymmetric Memory Interface," *JSSC*, vol. 44, no. 4, 2009.

[23] S. Li, J. Ahn, J. B. Brockman, and N. P. Jouppi, "McPAT 1.0: An Integrated Power, Area, and Timing Modeling Framework for Multicore Architectures," HP Labs, Tech. Rep. HPL-2009-206, 2009.

[24] S. Li, *et al.*, "McPAT: An Integrated Power, Area, and Timing Modeling Framework for Multicore and Manycore Architectures," in *MICRO 42: Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, 2009, pp. 469–480.

[25] S. Li, K. Chen, J. H. Ahn, J. B. Brockman, and N. P. Jouppi, "CACTI-P: Architecture-Level Modeling for SRAM-based Structures with Advanced Leakage Reduction Techniques," in *ICCAD*, 2011.

[26] S. Li, *et al.*, "A Heterogeneous Lightweight Multithreaded Architecture," in *International Parallel and Distributed Computing Computing Symposium (IPDPS), MTAAP workshop*, 2007.

[27] S. Li, S. Kuntz, J. Brockman, and P. Kogge, "Lightweight Chip Multi-Threading (LCMT): Maximizing Fine-Grained Parallelism On-Chip," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 7, July, 2011.

[28] S. Li, S. Kuntz, P. Kogge, and J. Brockman, "Memory Model Effects on Application Performance for a Lightweight Multithreaded Architecture," in *International Parallel and Distributed Computing Computing Symposium (IPDPS), MTAAP workshop*, 2008.

[29] X. Li, M. C. Huang, and K. Shen, "A Realistic Evaluation of Memory Hardware Errors and Software System Susceptibility," in *Proceedings of the 2010 USENIX conference on USENIX annual technical conference*, ser. USENIXATC'10, 2010, pp. 6–6.

[30] Los Alamos National Laboratory, Reliability Data Sets. [Online]. Available: {http://institutes.lanl.gov/data/fdata/}

[31] C.-K. Luk, *et al.*, "Pin: Building Customized Program Analysis Tools with Dynamic Instrumentation," in *PLDI*, Jun 2005.

[32] P. Rosenfeld et al, "DRAMSim2," http://www.ece.umd.edu/dramsim/.

[33] R. Palmer, *et al.*, "A 14mW 6.25Gb/s Transceiver in 90nm CMOS for Serial Chip-to-Chip Communications," in *ISSCC'07*, 2007, pp. 440–614.

[34] T. Rao and E. Fujiwara, *Error-Control Coding for Computer Systems.* Prentice Hall, 1989.

[35] A. F. Rodrigues, *et al.*, "The Structural Simulation Toolkit," *SIGMETRICS Perform. Eval. Rev.*, vol. 38, pp. 37–42, March 2011.

[36] Samsung Electronics Corporation, "Samsung Electronics Develops World's First Eight-Die Multi-Chip Package for Multimedia Cell Phones," 2005, (Press release from `http://www.samsung.com`).

[37] B. Schroeder and G. A. Gibson, "A Large-scale Study of Failures in High Performance Computing Systems," in *Proceedings of DSN*, 2006.

[38] B. Schroeder, E. Pinheiro, and W.-D. Weber, "DRAM Errors in The Wild: A Large-Scale Field Study," *Commun. ACM*, vol. 54, no. 2, pp. 100–107, 2011.

[39] Semiconductor Industries Association , "International Technology Roadmap for Semiconductors./ Model for Assessment of CMOS Technologies and Roadmaps (MASTAR) http://www.itrs.net/."

[40] H. Simon, "Exascale Challenges for the Computational Science Community," Lawrence Berkeley National Laboratory and UC Berkeley, Tech. Rep., Oct. 2010.

[41] C. Slayman, M. Ma, and S. Lindley, "Impact of Error Correction Code and Dynamic Memory Reconfiguration on High-Reliability/Low-Cost Server Memory," in *Proceedings of the IEEE Integrated Reliability Workshop*, 2006, pp. 190–193.

[42] B. J. Smith, "A Pipelined, Shared Resource MIMD Computer," in *Proceedings of the International Conference on Parallel Processing*, 1978, pp. 6–8.

[43] D. Strukov, "The Area And Latency Tradeoffs Of Binary Bit-Parallelbch Decoders For Prospective Nanoelectronicmemories," in *Proceedings of 2006 Asilomar Conference on Signals Systems and Computers*, Oct. 2006, pp. 1183–1187.

[44] Tezzaron Semiconductor, "Soft Errors in Electronic Memory–A White Paper," Tezzaron Semiconductor," Technical Report, 2004.

[45] A. N. Udipi *et al.*, "Rethinking DRAM Design and Organization for Energy-Constrained Multi-Cores," in *Proceedings of ISCA*, 2010.

[46] C. Wilkerson, *et al.*, "Reducing Cache Power With Low-Cost, Multi-Bit Error-Correcting Codes," in *International Symposium on Computer Architecture*, 2010, pp. 83–93.

[47] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The SPLASH-2 Programs: Characterization and Methodological Considerations," in *ISCA*, 1995.

[48] D. H. Yoon, *et al.*, "FREE-p: Protecting Non-Volatile Memory against both Hard and Soft Errors," in *Proc. the Int'l Symp. High-Performance Computer Architecture (HPCA)*, February 2011.

[49] J. W. Young, "A First Order Approximation To The Optimum Checkpoint Interval," *Commun. ACM*, vol. 17, pp. 530–531, September 1974.