Exploring DRAM Organizations for Energy-Efficient and Resilient Exascale Memories

Bharan Giridhar[‡], Michael Cieslak[‡], Deepankar Duggal[‡], Ronald Dreslinski[‡], Hsing Min Chen[†], Robert Patti[§], Betina Hold[*], Chaitali Chakrabarti[†], Trevor Mudge[‡], David Blaauw[‡]
[*]ARM Inc., San Jose, CA 95134
[†]Arizona State University, Tempe, AZ 85287
[§]Tezzaron Semiconductor, Naperville, IL 60563
[‡]University of Michigan, Ann Arbor, MI 48109
Contact: bharan@umich.edu

ABSTRACT

The power target for exascale supercomputing is 20MW, with about 30% budgeted for the memory subsystem. Commodity DRAMs will not satisfy this requirement. Additionally, the large number of memory chips (>10M) required will result in crippling failure rates. Although specialized DRAM memories have been reorganized to reduce power through 3D-stacking or row buffer resizing, their implications on fault tolerance have not been considered. We show that addressing reliability and energy is a co-optimization problem involving tradeoffs between error correction cost, access energy and refresh power—reducing the physical page size to decrease access energy increases the energy/area overhead of error resilience. Additionally, power can be reduced by optimizing bitline lengths. The proposed 3D-stacked memory uses a page size of 4kb and consumes 5.1pJ/bit based on simulations with NEK5000 benchmarks. Scaling to 100PB, the memory consumes 4.7MW at 100PB/s which, while well within the total power budget (20MW), is also error-resilient.

1. INTRODUCTION

The exascale supercomputing program has set a goal of producing an exaFLOP-class computer (capable of executing $>10^{18}$ FLoating point Operations Per Second) within a power budget of 20MW—today's supercomputers perform in petaFLOPS, consuming ~10MW [42]. GPGPUs have greatly improved processor efficiency in supercomputers [18], and emerging technologies such as near-threshold computing [12, 20] promise to improve this further. However, memory efficiency has been improving at a much slower rate, and exascale machines are projected to require at least 100PB of main memory capable of sustaining a bandwidth of 100PB/s (0.1B/FLOP) [37]. Main memory in today's petascale systems consumes ~30% of the total power [16], and projections show that a simple scaling of today's DDR3-based memory [25, 35] to 100PB will result in a power consumption of ~52MW. Thus, the memory subsystem by itself is equivalent to the power consumption of about 50,000 homes and far exceeds the 20MW power budget set for the entire system. Additionally, such a memory will suffer from soft and hard errors so frequently that rollback will take longer than the mean-time-to-failure [10].

DDR4, a newer JEDEC standard, only provides 2× improvement [35] in efficiency which also fails to meet this target. While mobile DRAM standards like LPDDR2/3 [13] provide larger improvements (6-7×), they offer 3-5× less bandwidth compared to DDR4, thus requiring a much larger number of chips for the same performance. This in turn increases their fault tolerance requirements. With 3D-stacking, DRAM chips can be integrated very tightly with logic dies. These logic dies can take over some of the operations of the memory controller and allow much of the internal organization details to be hidden from the processor. This opens up flexibility in defining the bitcell array configurations, such as the size of the row buffer. High-performance computing (HPC) memories such as the Hybrid Memory Cube (HMC) [35] have taken a step in this direction, showing an increasing willingness to move away from pre-existing DRAM standards. Recently, JEDEC also proposed the Wide I/O standard [13] for mobile DRAM that is also based on 3D-stacking and is projected to provide ~12× improvement in efficiency over DDR3 - however, like other mobile DRAMs, it offers 10× less bandwidth than HMC-class solutions.

Memory errors in current systems contribute more than 40% of the total hardware-related failures and are projected to further increase in exascale systems [28], making error correction capability increasingly important. Hence, any changes to the DRAM organization must consider the impact on error resilience. While several recent DRAM architectures have proposed reducing access energy through 3D-stacking [21, 31] and row buffer resizing/rank subsetting [6, 7, 41, 45], they have not considered their implications on fault tolerance, which needs to be included as an integral part of the power optimization problem.

The focus of this paper is energy and reliability for exascale memories. We co-optimize error resilience costs, access energy and refresh power to arrive at an energy-efficient and resilient 3D-stacked memory for exascale computing. In addition to its area and power advantage, 3D integration allows us to stack conventional bitcells fabricated in an existing DRAM technology (50 nm) over a 28nm CMOS logic die.

SC13 November 17-21, 2013, Denver, CO, USA
Copyright 2013 ACM 978-1-4503-2378-9/13/11 ...$15.00.
http://dx.doi.org/10.1145/2503210.2503215

This also limits the design risk to just the stacking technology (already demonstrated in commercial products) and is an alternative to more speculative low-power non-volatile memory technologies. In order to address power and reliability, we make the following key contributions:

1. **Reduce DRAM refresh power by restructuring subarrays to minimize bitline capacitance.** DRAM bitcells must be periodically refreshed in order to retain data and a number of efforts have been presented to reduce the associated power consumption [17,30]. In a 100PB memory built using DDR3 chips, refresh power alone can be as high as 3-4MW, consuming 20% of the total power budget for the system, even in standby mode. We optimize subarray column height (bits per bitline) to minimize the total bitline capacitance using a tradeoff between decreased local bitline capacitance and increased muxing/routing capacitance to reduce refresh power. This achieves ~4.6× savings in refresh power with a 9.7% increase in subarray area.

2. **Optimize the energy/area overhead of including stronger resilience mechanisms.** Traditional DIMM-based solutions use Chipkill [11] to protect against single DRAM chip failures. By analogy, we propose *Subarraykill*—a fault tolerance mechanism implemented on subarrays in a bank. The scheme protects against soft errors (occuring primarily due to particle strikes causing burst errors in a subarray), as well as hard errors (such as multi-bit faults along columns/rows and 3D technology-specific faults such as TSV failures). While most existing schemes use Single-bit Error Correction Double-bit Error Detection (SECDED) ECC in conjunction with Chipkill, we find that for smaller pages, such schemes significantly increase check-bit area/ power overheads. In this paper we propose using rotational Single Byte Error Correction Double Byte Error Detection (SBCDBD) ECC with 4-8b per byte[1] [36] to reduce these overheads. For instance, accessing a 128b data word in a 4kb page using a (144, 128) SBCDBD[2] (B=4b) ECC decoder instead of 4×(39, 32) SECDED decoders reduces access energy by 26% and check-bit storage and refresh power overheads from 21.9% to 12.5% without decreasing error coverage.

3. **Include the impact of data locality on the optimal page size.** A physical page consists of data from multiple subarrays of bitcells in a bank. Access energy primarily results from activating rows of bitcells with a RAS (Row Address Strobe). Reducing the page size decreases the energy spent per RAS by activating fewer subarrays. On the other hand, if workloads exhibit good data locality, larger pages are desirable as higher reuse of the page contents reduces the number of RASs and results in greater energy savings. We include this tradeoff in the optimization study by simulating our DRAM model with NEK5000 [2] benchmarks representing anticipated exascale applications.

Our solution is a 32Gb 3D-stacked DRAM with a page size of 4kb, access energy of 5.1pJ/bit and standby power of 0.75pW/bit. For 100PB, the total power consumption is ~4.7MW at a data bandwidth of 100PB/s. This is an improvement of ~6.5× over DDR4 DIMM-based solutions and ~1.8× over the first generation HMC. This leaves 15MW for processors, interconnect, cooling and the other sources of power losses in an exascale system.

2. BACKGROUND AND MOTIVATION

2.1 Power Challenge

Commodity DRAMs, primarily driven by volume, use very few data pins per chip to reduce packaging/test costs. As a consequence, such architectures open a page across multiple chips in a DIMM in order to meet bandwidth requirements. Every access opens an 8kb row buffer in each chip (constituting an 8kB page) in order to fetch a cache line, which is typically only 64B [7]. The spatial locality of single-core workloads takes advantage of this large row buffer; however, with reduced locality in future multi-cores [41] this over-fetch renders commodity DRAMs very energy-inefficient and unsuitable for exascale computing [16]. Thus, using today's DDR3-1333 chips (517.63mW/GB/s [35]), an exascale memory with a 100PB/s data bandwidth would consume ~52MW, far exceeding the 20MW target for the entire system. With DDR4-2667 chips (309.34mW/GB/s [35]), this decreases to 31MW, which also fails to meet this target. With mobile DRAMs like LPDDR2 (80mW/GB/s) and LPDDR3 (70mW/GB/s) [13] the total memory power further reduces to 8MW and 7MW respectively. However, they provide much less bandwidth (4.3-6.4GB/s) compared to DDR4 (21.34GB/s), thus requiring a much larger number of chips to achieve the same performance. This, in turn, impacts cost, area and fault tolerance requirements.

To address these issues, we explore 3D integration. In addition to its inherent area and power benefits, 3D-stacking also decouples bitcell and logic design, which are known to have contrasting process needs: (1) DRAM bitcells benefit from a low-leakage process for longer data retention; (2) the peripherals (such as sense amplifiers, wordline drivers and I/O) are designed for high speed and benefit from a high performance process. 3D-stacking allows DRAM designers to integrate these distinct processes, as well as reduce latency and increase bandwidth through the use of a large number of high-speed TSVs. The first generation HMC is one such example of a 3D-stacked memory that improves the power efficiency to 86.5mW/GB/s while providing a per-stack bandwidth of 128GB/s [35]. With over 7× power savings compared to DDR3, a 100PB main memory built using HMCs will consume ~8.8MW. While this is within the power budget of the whole system, it still amounts to a sizeable 44%—more than the 30% used as today's rule of thumb [16] for memory. Recently proposed Wide I/O mobile DRAM, also based on 3D-stacking, is projected to further improve the efficiency to 40mW/GB/s [13] (4MW for an exascale memory) - however, it only provides a bandwidth of 12.8GB/s (10× lesser than HMC) and also does not give any special consideration to error resilience.

DRAM memories also consume power due to refresh. With the 64ms refresh interval, a 1Gb DDR3-1333 chip consumes 4mW of refresh power [33]. An exascale main memory built using such chips will consume as much as 3-4MW for refresh alone, even in standby mode. More recently, several approaches [17, 30] have focused on optimizing refresh interval as a function of process and temperature variations. However, there is scope for additional power savings, orthogonal to such techniques, by reorganizing subarray layout.

2.2 Resilience Challenge

In addition to meeting stringent power constraints, large scale machines also require stronger mechanisms for error

[1] 'Byte' in ECC terminology represents a symbol of multiple bits, not necessarily the customary 8 bits.

[2] (144,128) represents 128 data bits and 16 check-bits for ECC, totaling 144 bits of storage.

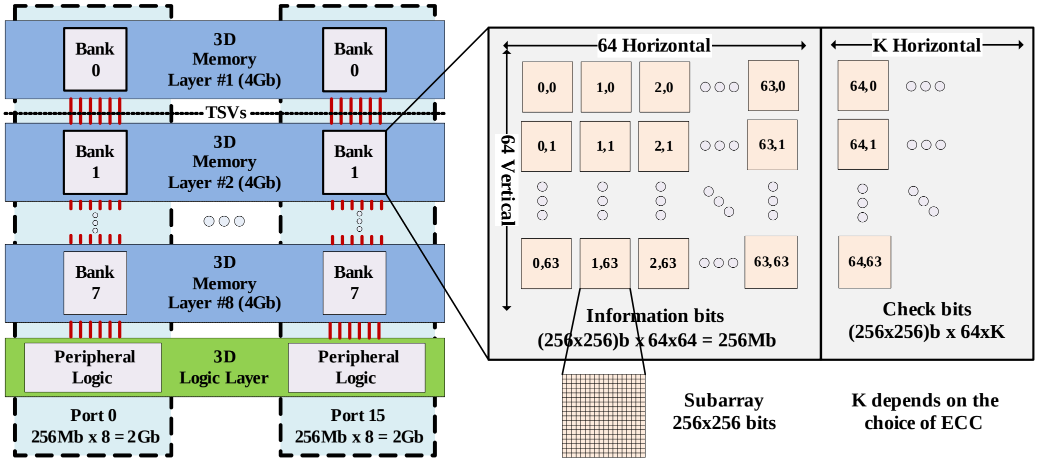

Figure 1: Logical organization of the 32Gb 3D-stacked DRAM. The DRAM capacity (32Gb) only accounts for information bits and does not include check-bit storage overhead, which depends on the choice of ECC.

resilience. Errors can be broadly classified as soft and hard. Soft errors are generated by energetic particle interactions with semiconductor devices and are transient in nature. Such particle strikes typically affect multiple bits and cause burst errors in modern DRAM processes. This failure mode has traditionally been considered the dominant fault mechanism in DRAM DIMMs. Errors are detected, and may also be corrected, using ECC. Commodity ECC DIMMs use a (72, 64) SECDED ECC to protect 64 bits of data using 8 check-bits and are constructed using 18 x4 chips (x4 ECC DIMM), or 9 x8 chips (x8 ECC DIMM) [44]. They use a 72b wide data path where the additional DRAM chips are used to store both information and check-bits. The 4 (or 8) bits coming from each chip are spread apart spatially such that no more than one bit is affected by a particle strike—allowing for SECDED ECC to correct such errors.

Hard errors, on the other hand, are related to manufacturing process variations and device wearout. They can be intermittent (data pattern dependent) or permanent in nature [39]. 3D-stacking technology introduces another source of hard errors in the form of TSV faults. Recent studies suggest hard errors are at least as significant as soft errors in large-scale systems [19, 38, 39]. Accordingly, most server-grade DIMMs use stronger protection mechanisms which can tolerate a whole-chip failure, known as Chipkill. One way of implementing Chipkill is by sending each data bit of a DRAM chip (e.g. each of the 4 bits of an x4 chip) to a separate ECC word (a set of data and check-bits over which the ECC algorithm performs error detection/correction). The other method of implementing Chipkill employs the use of stronger codes, such as SBCDBD ECC, that can correct multiple-bit errors. Such codes use Galois Field (GF) arithmetic with b-bit symbols (or bytes) to tolerate up to an entire memory chip failure. The (144,128) SBCDBD with 4b per byte is one such code which can correct up to 4 adjacent bit errors. This code has the same check-bit storage overhead as the (72, 64) SECDED ECC (12.5%), but has much higher detection and correction capability. This ECC scheme also has a highly parallel implementation, with at most 3-4 gates in the critical path. Thus the delay overhead of this scheme is <0.4ns in 28nm CMOS.

Although such mechanisms must ideally be able to correct for all such DRAM failure modes, this is not always possible. In such cases a rollback to the last checkpoint is performed (which incurs additional power and computation latency penalties). Finally, software intervention is required to retire pages with permanent failures [40].

3. PRELIMINARY 3D ARCHITECTURE

3.1 Basic organization

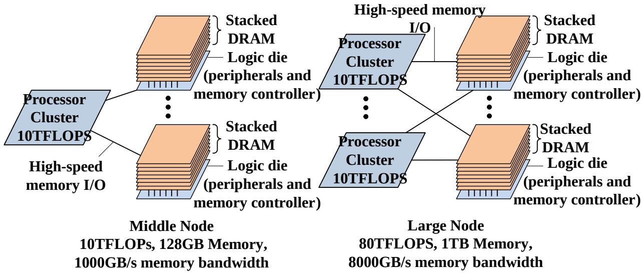

Figure 2: Middle and large node architectures for exascale computing [26].

We propose building a 100PB main memory using smaller 3D-stacked devices based on Tezzaron's 3D-stacking process [4] that allows us to stack 8 memory layers on a logic die through finely-spaced (1.75μm pitch) low-power TSVs. The TSVs have a feedthrough capacitance of 2-3fF and a series resistance of $<3\Omega$. This allows as much as 4GB of data in each individual stack. We assume a middle to large node architecture (Fig. 2) where these 3D-stacked devices will be connected to processor dies through high-speed I/O [26] and then multiple such nodes will be organized into racks across the entire system to make 100PB of total memory ($\sim 2.5\times10^7$ of these 3D chips in total in the overall system).

Building such a system has a number of design challenges. As a starting point, the focus of this paper is on the power and resilience of the building block—the 32Gb 3D-stacked DRAM. The 32Gb stack's logical organization is shown in Fig. 1. This will serve as the base architecture for co-optimizing energy and resilience. Note that the '32Gb' ca-

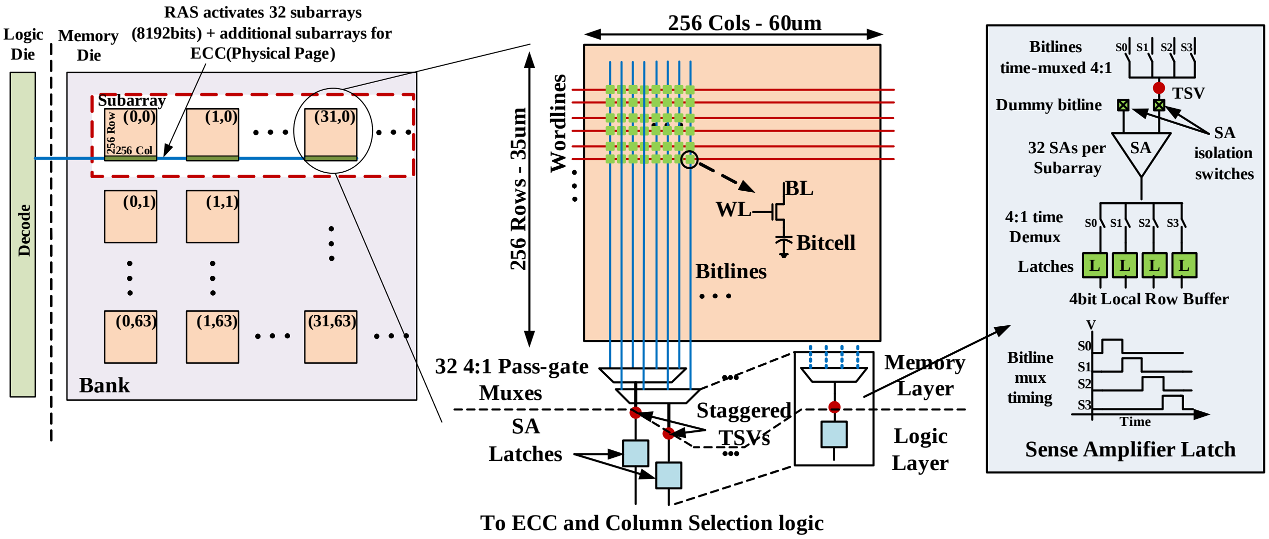

Figure 3: Diagram showing how RAS operations are performed in each bank. The bitlines are 4:1 time multiplexed to one TSV as the TSV pitch (1.75μm) is much larger than the bitline pitch (0.5μm).

pacity only accounts for data bits and does not include the check-bit storage area overhead which depends on the choice of ECC (discussed in Section 4). Each 32Gb 3D chip consists of 8 4Gb DRAM memory dies stacked on top of a logic die. The organization of the 4Gb DRAM die is based on Tezzaron's existing Octopus [3] DRAM solution. Each 3D stack has 16 128-bit data ports, with each port accessing an independent 2Gb address space. Each address space is further subdivided into 8 256Mb banks. Each bank, in turn, is physically organized as 64×64 matrix of subarrays (not including subarrays for storing ECC check-bits). Each subarray is a 256×256 arrangement of bitcells and is 60μm×35μm (Fig. 3).

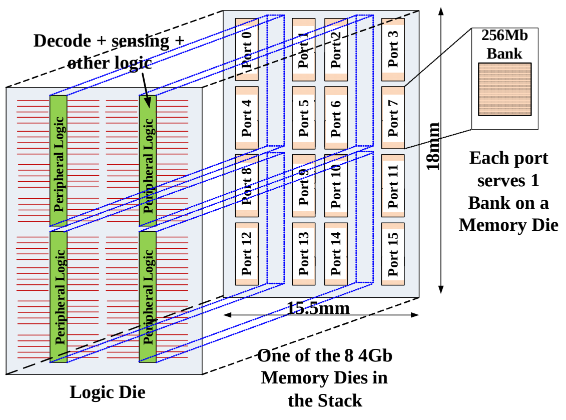

Figure 4: Physical floorplan of the logic die and a 4Gb memory die in the 3D stack (ignoring ECC overhead). The top-down view of the stack shows that DRAM banks are arranged around 'spines' of peripheral logic.

Figure 4 shows the physical floorplan of each 4Gb DRAM memory die and the logic die. The logic die is fabricated in a 28nm CMOS process and consists of address-decoding logic, global wordline drivers, sense amplifiers, row buffers, error correction logic and low-swing I/O logic with pads. Each memory die is partitioned into 16 ports with each port serving 1 of the 16 banks on a die (Fig. 4). The memory die is fabricated in a 50nm DRAM process and consists of the DRAM subarrays along with some logic such as local wordline drivers and pass-gate muxes. While there are more advanced DRAM processes (e.g. 20nm), TSV yield in existing 3D-stacked prototypes has only been proven up to the 50nm DRAM process node [23, 35]. All subarrays in a vertical stack share the same row buffer using TSVs and hence at most one row of subarrays in a vertical stack can have its contents in the row buffer, which corresponds to a physical page. Thus, assuming an 8kb page, a maximum of 2048 pages can be simultaneously open per device (128 8kb pages per bank × 16 banks per physical layer), providing concurrency similar to Sub-Array Level Parallelism [24]. The device provides a sustained bandwidth of 6.25GB/s per port (100GB/s total).

We now describe DRAM operations for this architecture for an 8kb page (information bits), which is the typical size today [7]. Note that for smaller page sizes, the decoding and other peripheral logic will change accordingly. DRAM operations are pipelined and each port can perform up to three tasks in each clock cycle. Hence, at each port, one bank could be doing a RAS while another one could be simultaneously doing a Column Address Strobe (CAS); and a third one could be doing a Page Close (PC). Thus, in each cycle, we can open 16 new pages, read/write 16 data words and close 16 other pages [3].

3.2 RAS Operation

During RAS (Fig. 3), the address is first decoded in the logic layer to determine the bank and the subarrays to be activated. The decoded address is sent through the TSVs to the corresponding memory die containing the row of subarrays to be accessed. This ultimately activates one wordline in 32 subarrays out of the 64 subarrays in a row, (i.e. 32×256 = 8192 bitcells), causing them to charge share with their corresponding bitlines. In an error-resilient memory with ECC additional subarrays will have to be activated due to the check-bit storage overhead. The final charge-shared values are sensed using sense amplifiers on the logic layer (through TSVs) and the data is latched onto the row buffer (also on the logic layer) and a page is opened.

3.2.1 Sensing Scheme

In a conventional DRAM memory, the bitlines are initially precharged to VDD/2 and are allowed to charge share with the accessed bitcells. The sense amplifier then compares the new bitline voltage with a dummy/reference bitline at VDD/2 and stores this value onto a latch. The sense amplifier also causes a full swing on the bitline, thereby reenforcing the bitcell data. Since the TSV pitch (1.75μm) is much larger than the bitline pitch (~0.24μm), we stagger the TSVs and employ time-multiplexing on the bitlines to limit the number of TSVs and meet the minimum TSV pitch requirement. Note that the keep-out area around the TSVs is 500nm, allowing logic to be very close to the TSVs. The bitlines are 4-way time-multiplexed to one sense amplifier. The mux output drives a TSV which also serves as a global bitline vertically running through the stacked memory layers. The global bitline terminates on the logic die where it drives a sense amplifier and sensed data is stored into a 4-bit latch local to this amplifier (Fig. 3, SA Latch). This latch serves as a row buffer local to the sense amplifier. An 8kb row buffer consists of 2048 SA Latch units.

During every RAS, the sense amplifier evaluates 4 times in order to fill the row buffer with the corresponding bitcell values. Since full swing on the bitline involves charging/discharging large capacitances (~100fF), the standard sensing scheme is slow and does not meet our high speed sensing requirement. We make an optimization by decoupling the sense amplifier from the bitlines during evaluation using isolation switches (Fig. 3) in order to boost the sensing speed. Additionally, this also allows the sense amplifier to be reset more quickly before the next firing. The RAS latency (t_{RCD}) with such a sensing scheme is only 5ns [3] compared to 13.5ns in DDR3-1333 chips [1]. Unlike conventional schemes, the full swing on the bitlines is deferred and performed during the next Page Close (PC). This is acceptable as the data is available in the row buffer throughout this period. The shorter RAS latency also offsets the longer PC (also 5ns).

3.3 CAS Operation

During CAS, data is moved from the row buffer to the I/O logic. The address bits are decoded to select the relevant 128 bits from the row buffer to be sent out to the I/O port. Multiple CAS operations can be performed per RAS as long as the access is to the row already stored in the buffer. In order to protect against errors, ECC decoding and correction is also performed in this phase (Section 4). The corrected data is then sent to the I/O port using a low swing interface consuming 1pJ/bit at 1GT/s [3,14]. The CAS read latency is 2.5ns. Finally, during PC, the row buffer contents are written back to the subarray row and the buffer is released.

3.4 Access Energy Reduction

Table 1 illustrates how RAS energy dominates the total access energy per bit; it assumes that every access results in both a CAS and a RAS. There are two competing approaches to reduce total RAS energy—(1) reduce the total number of RAS operations; or, (2) reduce the energy per RAS operation. While large page sizes suffer from higher energy per RAS, spatial data locality will reduce the number of RASs by increasing the reuse of page contents. In contrast a smaller page size (activating fewer subarrays per RAS) reduces the energy per RAS, but suffers more RAS operations due to loss of data locality. Further complicating the tradeoff is that smaller pages tend to have higher ECC check-bit storage/power overhead (Section 4), which also increases access energy. We use these tradeoffs to arrive at the final, energy-optimal page size in Section 6.1.

RAS+PC Energy (pJ/bit)	12.7
CAS Energy (pJ/bit)	2.85
Low-swing I/O Energy (pJ/bit)	1.00
Total Access Energy (pJ/bit)	16.6

Table 1: Initial break-down of access energy in the 3D DRAM architecture for an 8kb page size (no optimizations included).

3.5 Refresh Power Reduction

Tezzaron's 256 × 256 subarray has total bitline load of ~100fF, bitcell capacitance of 25fF and VDD of 1.2V. Each bitcell needs to be refreshed within 64ms—a common DRAM standard. This implies that if a bitcell is read at the end of 64ms (just before a refresh), the final charge-shared voltage on the bitline, $V_{chargeshare}$ should still be sensed correctly. Typically, the sense amplifier requires a certain voltage margin between the sensed and the dummy bitline (at VDD/2 = 0.6V) in order to guarantee correct operation. This is around 100mV accounting for process variation and mismatch. Thus the minimum $V_{chargeshare}$ for sensing a '1' in our scheme is 0.7V.

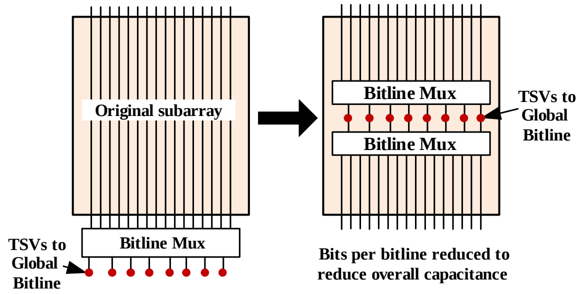

Figure 5: Reorganizing subarrays to shorted local bitlines and reduce refresh power.

If the subarray layout is changed such that the number of bits on a bitline is reduced (Fig. 5), the total capacitance charge-sharing with the bitcell during reads can be reduced. With reduced charge-sharing, $V_{chargeshare}$ retains a higher voltage value and hence the bitcell can also leak longer while meeting the minimum voltage requirement on $V_{chargeshare}$, before requiring refresh. This concept can be used to reduce the refresh power by (1) reducing the total capacitance charging/discharging during refresh, and (2) increasing the mean interval between refreshes. This technique is also effective in reducing variations in required refresh intervals among the bitcells, because charge-sharing between the bit-

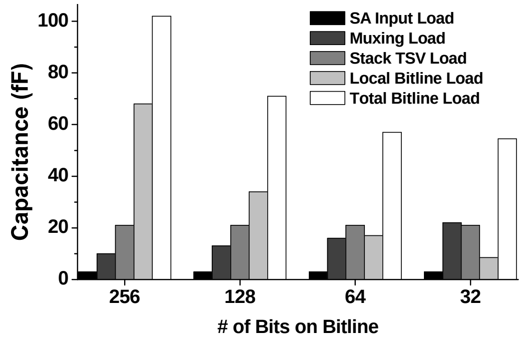

Figure 6: Components of charge-sharing capacitance as a function of number of bits on a bitline. While bitline capacitance reduces, the muxing capacitance increases.

cell and the bitline is linearly proportional to the bitline capacitance, thus the mean as well as the standard deviation of the refresh interval distribution improve linearly.

However, as the number of bits per bitline reduces, the capacitance of muxing (including routing through TSVs) increases (Fig. 6) which eventually negates further refresh savings (Fig. 7). Also, the increased area overhead of the extra TSVs and bitline muxing also reduces the subarray area efficiency. We choose to employ 64 bits on a bitline to get a ~4.6× savings in refresh power. This increases the subarray area by 9.7% due to additional TSVs and muxing. Note that reducing bitline capacitance also reduces the energy spent on RAS during an access. For an 8kb page, moving to 64 bits on a bitline (from 256 bits) reduces RAS energy from 12.7pJ/bit to 8.6pJ/bit.

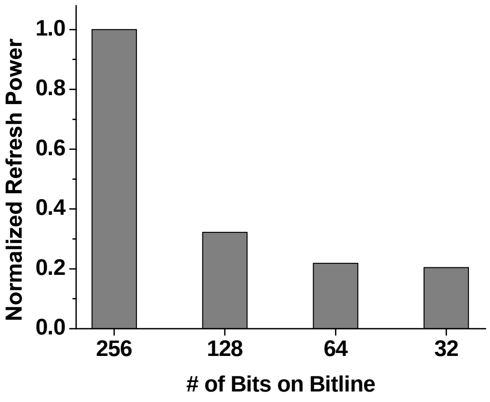

Figure 7: Refresh power savings as a function of number of bits on a bitline. The increase in muxing and routing power offsets the savings in bitline swing power as we move to 32 or fewer bits on a bitline.

4. 3D MEMORY WITH *SUBARRAYKILL*

In this section we propose Subarraykill—a resilience approach intended to protect against soft errors caused by particle strikes and hard errors caused by multiple errors along rows and/or columns in a subarray, referred to as subarray failures. This is analogous to how typical server-grade memory uses Chipkill [11] to protect against single memory chip failures or multiple errors from any portion of a single memory chip on a DIMM. We include hard error protection, because, as noted earlier, hard errors are at least as significant as soft errors [19, 38, 39]. In particular, row failures are the most frequent [29]. Subarraykill is implemented by spreading a data word evenly across a whole accessed page in order to tolerate multiple errors in the same subarray. Such errors can be detected and corrected using either SECDED or SBCDBD-based codes, which we compare in the next two sections. In summary, Subarraykill is designed to handle multi-bit faults, column faults and row faults (up to whole subarrays). In addition, since the ECC is on the logic die at the base of the 3D stack, we also protect against TSV failures.

4.1 SECDED-based Subarraykill

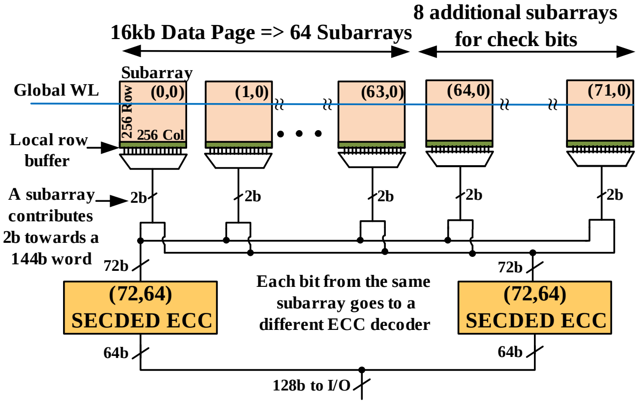
16kb Data Page => 64 Subarrays
8 additional subarrays for check bits
Global WL
Subarray
(0,0)
256 Row
256 Col
(1,0)
(63,0)
(64,0)
(71,0)
Local row buffer
A subarray contributes 2b towards a 144b word
2b
72b
(72,64) SECDED ECC
Each bit from the same subarray goes to a different ECC decoder
64b
128b to I/O

Figure 8: SECDED-based Subarraykill error correction for 16kb page.

We discuss the SECDED-based scheme using a 16kb page as an example as shown in Fig. 8. Borucki et al. [9] showed that a particle strike can affect almost 20 bits in the 110nm process. Since our DRAM uses the 50nm process, we conservatively protect against burst errors of up to 64 bits due to particle strikes. Additionally, we must also protect against hard errors such as row failures. In order to correct burst errors due to particle strikes and row failures, the 144-bit code word is spread across 64+8 subarrays and each subarray only contributes 2 bits towards the data word. During a read, each of these 2 bits is sent to a separate ECC unit so that in case of a soft error, or a subarray failure, each ECC word can have at most 1 bit in error which can be corrected using a conventional (72,64) SECDED code.

Table 2 shows that as the page size is reduced, fewer subarrays get activated and more bits are pulled from each subarray to form a word. With the SECDED scheme, these bits need to be sent to separate ECC words for decoding. Thus, as the page size is reduced, the SECDED code also becomes smaller which increases the storage overhead due to check bits. Table 2 also shows the storage overhead of error correction as a function of page size. Note that for a page size of 4kb, storage overhead with SECDED-based Subarraykill is 37.5%, which reduces the useful DRAM density to an unacceptable amount. This increase in storage overhead in turn increases the energy spent in retrieving these check bits during access and more prominently during refresh (Fig. 10). So, we propose switching to SBCDBD-based

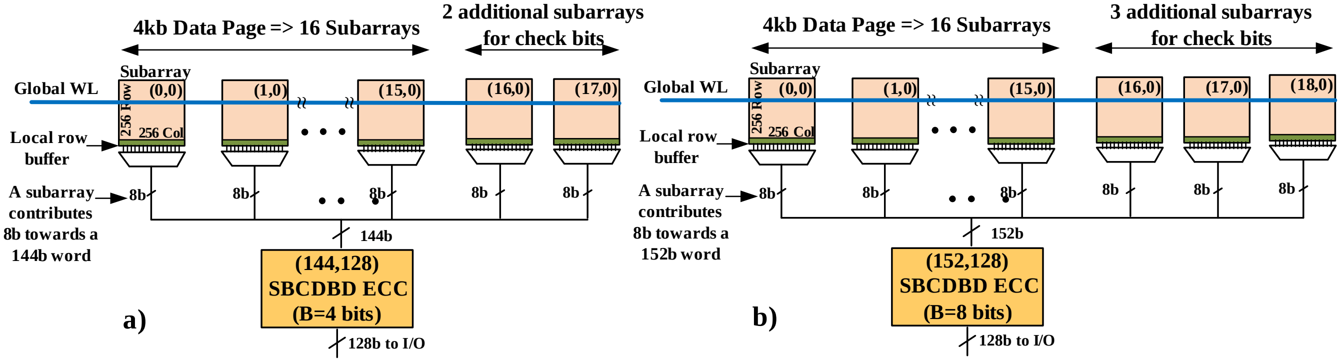
4kb Data Page => 16 Subarrays
2 additional subarrays for check bits
Subarray
Global WL
(0,0)
(1,0)
(15,0)
(16,0)
(17,0)
256 Row
256 Col
Local row buffer
A subarray contributes 8b towards a 144b word
8b
144b
(144,128) SBCDBD ECC (B=4 bits)
128b to I/O
a)
4kb Data Page => 16 Subarrays
3 additional subarrays for check bits
Subarray
Global WL
(0,0)
(1,0)
(15,0)
(16,0)
(17,0)
(18,0)
256 Row
256 Col
Local row buffer
A subarray contributes 8b towards a 152b word
8b
152b
(152,128) SBCDBD ECC (B=8 bits)
128b to I/O
b)

Figure 9: SBCDBD-based Subarraykill error correction options for 4kb page (information bits).

Page size (info. bits)	# accessed data arrays	# sel. bits per subarray	SECDED Config.	SECDED Check-bit Overhead	SECDED Latency Overhead in 28nm	SBCDBD Config.	SBCDBD Check-bit Overhead	SBCDBD Latency Overhead in 28nm
16kb	64	2	2×(72,64)	12.5%	0.62ns	1×(144,128), (B=4b)	12.5%	0.88ns
8kb	32	4	4×(39,32)	21.9%	0.49ns	1×(144,128), (B=4b)	12.5%	0.88ns
4kb	16	8	4×(39,32)* 8×(22,16)	21.9%* 37.5%	0.49ns* 0.38ns	1×(144,128), (B=4b)* 1×(152,128), (B=8b)	12.5%* 18.75%	0.88ns* 1.08ns
2kb	8	16	8×(22,16)*	37.5%*	0.38ns*	1×(152,128), (B=8b)*	18.75%*	1.08ns*
*code guarantees hard error detection (not correction) with soft error correction.								

Table 2: Comparing Subarraykill configurations for accessing a 128b information word from different page sizes using SECDED and SBCDBD ECC codes that have the same error correction performance.

ECC codes which have lower energy and storage overheads as will be discussed in the next section.

4.2 SBCDBD-based Subarraykill

We noted earlier (Section 3.4) that opening smaller pages directly improves total access energy, because it is dominated by RAS. However, SECDED-based Subarraykill incurs a high energy and area overhead for smaller page sizes, negating the advantages of smaller pages. As a solution, we propose using rotational SBCDBD-based codes which have lower area and energy overheads for the same error correction performance. Rotational codes have the additional merit that hardware implementation has low latency overheads (Table 2).

Consider the 4kb page (information bits) as shown in Fig. 9. The page is opened across 16+K subarrays (where K is the number of additional subarrays to store check bits) and the 144bit word is spread across this page with 32b spacing. This implies that each subarray supplies 8 bits towards the word. Because the bits are spaced 32 bits apart, this ensures that a particle strike would not cause a burst error of greater than 2 bits. Using a (144,128) rotational SBCDBD (B=4b) code [36], as shown in Fig. 9a, will ensure that no more than one 'byte' is in error, and hence can be corrected. Thus we have reduced the check bit storage area overhead to 12.5% (versus 21.9% with SECDED).

While the scheme explained in Fig. 9a protects against soft errors, it cannot guarantee correction in case of hard errors, such as whole subarray row failure, although it will detect them. This is because a subarray row failure would corrupt up to 8bits, or 2 bytes for B=4b. The (144,128) SBCDBD code with B=4b can detect errors spanning 2 bytes but cannot correct them.

If we need to guarantee correction in the case of subarray row failures, stronger SBCDBD codes are required at the cost of higher area, power and latency overheads. For example, we can modify the scheme from Fig. 9a to Fig. 9b and use a (152,128) SBCDBD (B=8b) ECC unit instead of a (144,128) SBCDBD (B=4b) ECC unit. The (152,128) SBCDBD (B=8b) ECC is a shortened RS(19,16) code over GF(2^8) with minimum distance 4 that is derived from RS (255,252) code over GF(2^8). Any subarray row failure will now affect at most 1 byte (8b) in an ECC word, and can be corrected. Table 2 compares SBCDBD configurations for different page sizes with their SECDED counterparts that have the same error correction performance. For instance, for the 4kb page (information bits), the (152, 128) SBCDBD (B=8b) ECC has 18.75% storage overhead compared to the 37.5% for the SECDED configuration and can correct whole subarray failures. While the reduction in storage overhead is significant, in memory systems this may still be too large. In such cases, we propose the use of the (144,128) code which has a 12.5% storage overhead and guarantees soft error correction, but only hard error detection. These configurations have been labeled with a '*' in the table. We end our analysis with the 2kb page size, as with smaller pages this storage overhead becomes prohibitive, increasing die cost.

Now consider the case when multiple subarrays have hard and soft errors. In other words, there are more than 2 bytes of errors in the code words that are being processed by the SBCDBD units. The mentioned SBCDBD codes have more than a 99% probability of detecting 3 or 4 bytes of errors. In contrast, the Hsiao code, which is a popular (72, 64) SECDED code, has a 43% probability of triple error detection and more than a 99% probability of quadruple error detection [36]. Thus, the use of the SBCDBD codes allows

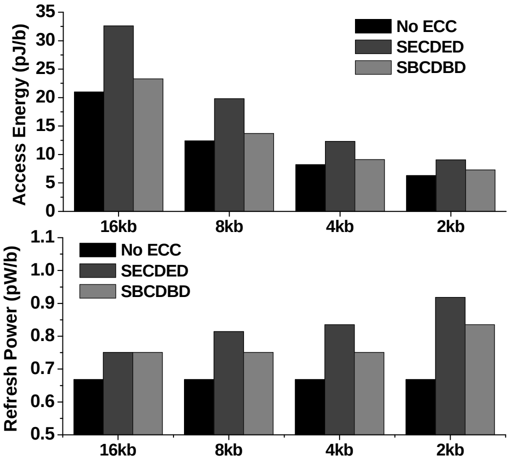

Figure 10: Comparing the impact of error correction on access energy and refresh power across page sizes.

us to detect more errors reliably and flag them for further processing by the OS. Even in the event of more severe faults (multiple subarrays, whole-chip, memory stack), we can still detect such failures using the same scheme by OR-ing the 'error-detect' signal [36] across subarrays.

5. EVALUATION METHODOLOGY

5.1 Power Model

The logic die model accounts for the power of the address decoders, sense amplifiers, row buffers and the low-swing I/O. The memory die model accounts for the power of bitlines and the global/local wordline drivers. All mixed-signal peripherals (sense amplifiers, precharge logic, etc.) were custom-designed in a 28nm industrial process and simulated in SPICE. All other logic layer components were synthesized in 28nm and the power was modeled using Synopsys Primetime$^{\copyright}$. All wire loads (including TSV routing capacitances) were estimated using the floorplan in Fig. 4. The sense amplifiers (along with bitline columns) were simulated across process, voltage and temperature. We performed 1M Monte Carlo simulations to design for process mismatch. The subarray is in a 50nm DRAM process and the model was provided by Tezzaron. The subarray model included the capacitances of the bitcell, the bitline and column muxing, as well as bitcell leakage current for accurate power modeling.

5.2 Performance Model

In order to obtain DRAM access patterns in large scale systems, we used main memory access traces from NEK5000 benchmarks [2]. These benchmarks characterize the applications expected in exascale machines. The benchmarks use a Message Passing Interface (MPI) [15] to communicate between cores. The individual benchmarks are described in Table 3. Since the total access latency t_{RCD} (RAS latency)+t_{RP} (Precharge latency)+t_{CL} (CAS latency) of the proposed 3D-stacked chip, including ECC enhancements, is much less than that of conventional DIMMs (13.6ns vs. 45ns for DDR3-1333 [1]), there is no negative impact on the total system runtime. Thus, the focus of our performance analysis is on the impact of row buffer locality on energy.

Benchmark	Description	Threads
blasius	Boundary layer analysis	64
conj_ht	2D example of conjugate heat transfer	64
eddy	Navier-Stokes Equation solver	64
ext-cyl	Flow past a cylinder in 2D	64
lowMach_test	Chemically reactive flow	64
solid	Linear elasticity steady 3D solid solver	24
axi	Axisymmetric boundary case	8

Table 3: NEK5000 benchmarks used for this study.

	L1	L2
Size	64KB	16MB
Block Size	64B	64B
Associativity	4	16
Replacement	LRU	LRU
Prefetch Policy	None	None
Write Back	Yes	Yes

Table 4: Cache parameters.

While it is impractical to evaluate an entire exascale system, we develop a methodology to simulate a single 32Gb 3D DRAM chip that is part of a blade unit in a larger exascale system. This is accomplished by running the benchmarks on a cluster with up to 64 cores per benchmark. Locality measurements are gathered on the access pattern for each benchmark. Additional traffic from other blades in the system would only serve to reduce the row buffer hit rate and, as such, the results presented here are optimistic for the amount of locality that will be experienced in exascale systems.

The evaluation cluster consisted of twelve 6-core Intel Xeon X5670 CPU nodes providing a total of 72 total cores. Each node contained 24GB of RAM and is connected to the network using 10Gbps Ethernet. We instrumented all reads / writes of the benchmarks with PIN [32]. PIN traces all memory accesses and instructions on each of the cores. Each process filtered its memory accesses through an L1 cache simulator before writing L1 misses out to a merged trace file. This merged file was run through an L2 cache simulator to generate a memory trace. The cache parameters are presented in Table 4.

Finally, this memory trace was run through a memory simulator to yield cache line locality in the memory row buffers. The memory simulator has two memory controllers: one that issues requests in-order and one that issues requests out-of-order with a scheduling window of 10. Since the out-of-order scheduler does not have a concept of time between accesses, the simulator gives the best case performance for locality with a scheduling window of size 10. We assume a standard DDR channel width of 128 bits and every memory access results in 4 consecutive DRAM accesses (burst of four CAS instructions) to fill a cache line (64B). On a subsequent access, if the row buffers already contain the data correlating to that address, we have a hit and can avoid a RAS access. On a row buffer miss a RAS is required in addition to a CAS. We studied the percentage of main memory accesses that resulted in a RAS as a function of row buffer (DRAM page) size in order to understand the impact of page size on locality.

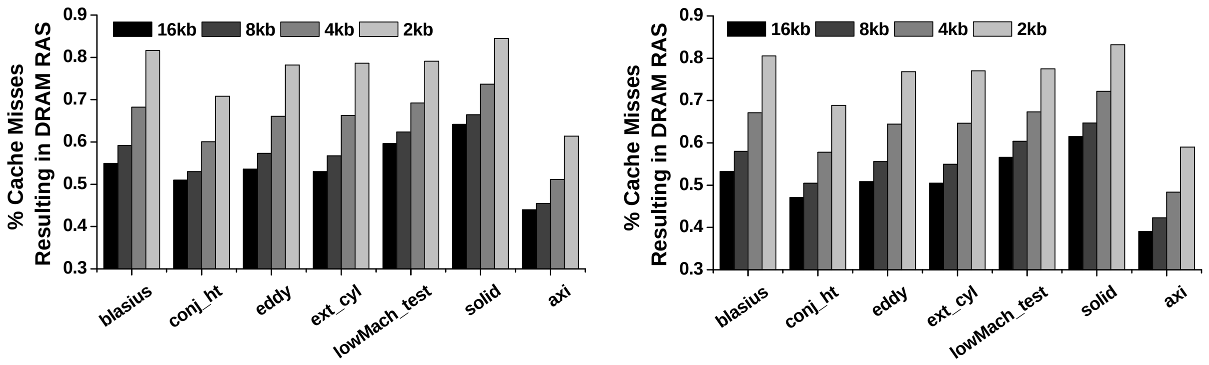

Figure 11: Percentage of cache misses resulting in a DRAM RAS operation with in-order (left) and optimistic out-of-order (right) memory scheduling.

6. RESULTS

The results section is broken into two parts. First we include the impact of benchmark locality on page size, and then we explore the total power consumption of our proposed design for a 100PB system, comparing it to conventional DIMMs and current 3D DRAM designs.

6.1 Locality-Resilience Tradeoff

Figure 11 shows the percentage of cache misses that result in a DRAM RAS operation across several NEK5000 benchmarks with (a) in-order and (b) out-of-order (window size 10) memory scheduling. As the page size becomes smaller, it is less likely that a subsequent access will hit the row buffer (lower locality). Overall, the aggregate locality and its dependence on page size is at best moderate, because multiple cores interleave accesses to the DRAM, evicting each others' data from the row buffer. Even with optimistic memory scheduling, the trend barely changes, and at 2kb (information page size), almost 80% of DRAM cache line accesses result in a RAS operation.

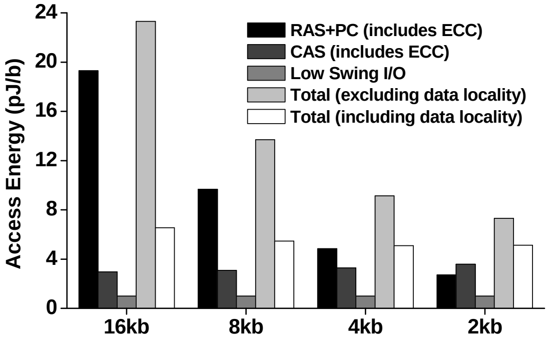

Figure 12: DRAM access energy across page sizes. This includes the tradeoff between locality and the energy cost of error correction using the proposed SBCDBD-based Subarrakill schemes.

The low locality of these benchmarks suggests using smaller pages to minimize energy. However, smaller pages have higher energy/area costs of error resilience. Taking locality and error resilience costs jointly into consideration, we minimize the energy of 3D DRAM chip. The storage-optimal ECC scheme for each page size is selected from the configurations in Table 2. The power/energy numbers were calculated with the methodology described in Section 5. Figure 12 shows the energy of DRAM accesses for each page size. The energy is broken down by RAS+PC energy and CAS energy that includes all ECC overheads, and low swing I/O. The fourth bar shows the total energy per bit without considering locality—all access result in a RAS+PC and a CAS. The locality of the benchmarks reduces the likelihood of an access requiring a RAS+PC, so the last bar shows a reduction resulting from the average locality across the benchmarks. For the 8kb page size, the locality helps to reduce the average access energy per bit from 13.7pJ to 5.5pJ. It is important to note that each cache line miss that results in a RAS operation performs 4 CAS operations to return the entire 64B cache line—this improves the locality by 4×. The total access energy of the system experiences a slight minima at the 4kb page size. Smaller page sizes incur slightly higher energy due to poor locality and additional area overhead for parity bit storage. The 4kb page has a low energy of 5.1pJ/bit with a storage overhead of 12.5% for ECC checkbits. The 12.5% overhead is quite common in today's ECC DIMMs [44]. Note that this particular implementation only guarantees hard error detection—if hard error correction is required, the 4kb page consumes 5.4pJ/bit with 18.75% storage overhead.

6.2 Complete 100PB DRAM Power Analysis

Taking into consideration the effect of locality on access energy, we calculate the total power of the exascale memory. To do this we scale the energy numbers to a 100PB memory with 100PB/s data bandwidth and include the power from refresh and error detection. Overall, we find that the total power of the DRAM system scales well to the 4kb page size (Fig. 13) without any additional ECC storage overhead (Table 2). The total power of the DRAM in the exascale system is 4.7MW (with hard error detection). This increases to 4.9MW if hard error correction is required (with 18.75% storage overhead). An exascale system built with conventional DDR3 DIMMs would consume ~52MW and one built with a 3D design such as the HMC would consume ~8.8MW. Overall, our proposed design results in a 6.5× reduction in power compared to DDR4 DIMMs, and a 1.8× reduction compared to the first generation HMC. We compare our solution with current DIMM-based and 3D-stacked DRAM memories in Table 5.

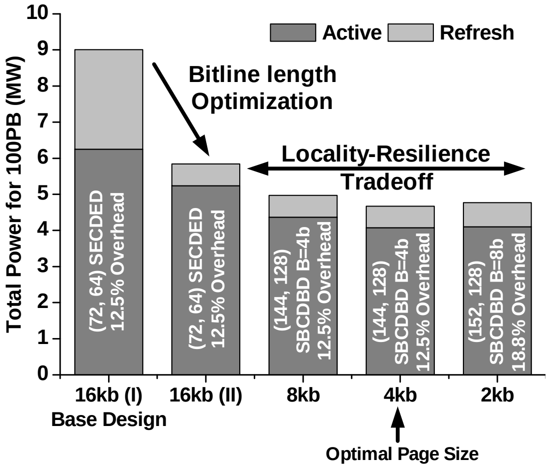

Figure 13: Power consumption of 100PB DRAM constructed using 32Gb 3D-stacked chips. Bitline length optimization reduces power by 35%. The locality-resilience co-optimization further reduces power by an additional 20%.

DRAM Memory	BW (GB/s)	Configuration	Power at 100PB/s
DDR3-1333 DIMM [35]	10.66	5E7× 2GB	52MW
DDR4-2667 DIMM [35]	21.34	2.5E7× 4GB	31MW
LPDDR3 [8] (30nm)	6.4	2E8× 512MB	6.3MW
HMC I [35] (3D-stack, 50nm DRAM, 90nm logic)	128	2E8× 512MB	8.6MW
Wide I/O [23] (3D-stack, 50nm)	12.8	2E8× 512MB	2.6MW*
This work (3D-stack, 50nm DRAM, 28nm logic)	100	2.5E7× 4GB	4.7MW
*Assumes 100% locality in a 16kb page.			

Table 5: Comparison with current DIMM-based and 3D-stacked DRAM memories.

7. RELATED WORK

DRAM Reorganization and Active Power: Several works have proposed reorganizing DRAM through rank sub-setting and subarray reorganization [7, 41, 45] and analyzed their impact on performance, power and reliability [5]. Our work focuses on reorganizing DRAM for an exascale system where we are constrained by stringent requirements on power, performance and reliability. Udipi et al. [41] reorganized the DRAM to enable single access filling of a 64B cache line. This approach reduced power consumption while trading off transfer times by having lower bandwidth. 3D-stacking enables us to have increased transfer bandwidth (through a large number of TSVs) and hence we did not have to make any major tradeoffs while scaling back on power. Work by Loh [31] on 3D DRAM architecture has mainly focused on design points involving multiple memory controllers and ranks; our reorganization is on a finer micro-architectural granularity. In [24], Kim et al. exploited parallelism in DRAM sub-arrays to reduce latency by overlapping memory accesses to different banks and reduce power by operating at a subarray granularity. More recently, Weis et al. [43] also performed a design space exploration for 3D-stacked DRAM, where they co-optimized the memory and the controller architecture to minimize energy. However, both approaches have not considered implications on error resilience costs.

Refresh Power: Recently, there have been attempts to reduce refresh rates as low as possible [17, 30] without introducing errors. These proposals make the observation that process variations and temperature conditions result in each DRAM cell having a different refresh rate. Accordingly, they refresh different portions of the DRAM at different rates after profiling the impact of different sources of variation. In addition, the controller can smartly avoid refreshing inactive DRAM rows. Both of these techniques to reduce refresh power are orthogonal to the technique presented in this paper and could be used to further reduce refresh power in future DRAM organizations. However, they would incur more storage and computation overheads to track DRAM cell refresh rates or the inactive rows. Recently Lee et al. [27] also proposed tiering DRAM bitlines to reduce capacitance and improve latency. However, they have not considered its implications for refresh power savings.

Error Correction cost: Several works [6, 22] have evaluated the cost of error correction on the overall system in the context of soft errors. However, more recent studies [19, 38] have shown that DRAM failures in large scale systems are dominated by hard errors. We implement Subarraykill to protect against hard errors such as whole-subarray failures. In addition, large scale systems use a memory scrubber that periodically walks through the memory and corrects the data with an ECC mechanism [34].

8. CONCLUSIONS

In this paper, we showed that addressing reliability while meeting exascale power budgets is a co-optimization problem involving multiple aspects of memory design. Contrary to popular belief, we showed that an all-DRAM solution is feasible, provided the traditional interfaces are rethought. We proposed a resilient architecture for a main memory building block based on a Tezzaron prototype. It employs a 3D-stacked DRAM organization to meet the stringent power, bandwidth and reliability demands. We showed that minimizing power requires a tradeoff between row buffer size, refresh, and the ECC mechanisms.

Using NEK5000 benchmarks, we showed that the optimal solution for the 32Gb 3D-stacked building block uses a 4kb page with (144,128) SBCDBD (B=4b) ECC scheme to protect against soft/hard errors. Scaling this design to 100PB would result in a memory power consumption of 4.7MW, which is well within the exascale power budget (20MW).

9. ACKNOWLEDGMENTS

We thank Shamik Ganguly for his help with simulations while at the University of Michigan. This work was partially sponsored by the Defense Advanced Research Projects Agency (DARPA) under agreement HR0011-13-2-0006, the Department of Energy (DOE) under the Advanced Scientific Computing Research (ASCR) program, and the National Science Foundation (NSF) grant nos. CNS-0910699 and CNS-0910851.

10. REFERENCES

[1] Micron 1Gb DDR3 SDRAM Datasheet. http://www.micron.com/.

[2] NEK5000. http://nek5000.mcs.anl.gov/.

[3] Octopus 8-Port DRAM for Die-Stack Applications. http://www.tezzaron.com/.

[4] Tezzaron Semiconductor. http://www.tezzaron.com/.

[5] J. H. Ahn, N. P. Jouppi, C. Kozyrakis, J. Leverich, and R. S. Schreiber. Future scaling of processor-memory interfaces. In *IEEE International Conference on High Performance Computing Networking, Storage and Analysis (SC)*, pages 1–12, 2009.

[6] J. H. Ahn, N. P. Jouppi, C. Kozyrakis, J. Leverich, and R. S. Schreiber. Improving System Energy Efficiency with Memory Rank Subsetting. *ACM Transactions on Architecture and Code Optimization (TACO)*, 9(1):4, 2012.

[7] J. H. Ahn, J. Leverich, R. S. Schreiber, and N. P. Jouppi. Multicore DIMM: an Energy Efficient Memory Module with Independently Controlled DRAMs. *Computer Architecture Letters*, 8(1):5–8, 2009.

[8] Y.-C. Bae, J.-Y. Park, S. J. Rhee, S. B. Ko, Y. Jeong, K.-S. Noh, Y. Son, J. Youn, Y. Chu, H. Cho, et al. A 1.2 V 30nm 1.6 Gb/s/pin 4Gb LPDDR3 SDRAM with input skew calibration and enhanced control scheme. In *IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pages 44–46, 2012.

[9] L. Borucki, G. Schindlbeck, and C. Slayman. Comparison of accelerated DRAM soft error rates measured at component and system level. In *IEEE International Reliability Physics Symposium (IRPS)*, pages 482–487, 2008.

[10] F. Cappello, A. Geist, B. Gropp, L. Kale, B. Kramer, and M. Snir. Toward Exascale Resilience. *International Journal of High Performance Computing Applications*, 23(4):374–388, 2009.

[11] T. J. Dell. A White Paper on the Benefits of Chipkill-Correct ECC for PC Server Main Memory. *IBM Microelectronics Division*, pages 1–23, 1997.

[12] R. G. Dreslinski, M. Wieckowski, D. Blaauw, D. Sylvester, and T. Mudge. Near-Threshold Computing: Reclaiming Moore's Law Through Energy Efficient Integrated Circuits. *Proceedings of the IEEE*, 98(2):253–266, 2010.

[13] S. Dumas. Mobile Memory Forum: LPDDR3 and WideIO. In *JEDEC Mobile Forum*, 2011.

[14] K. Fukuda, H. Yamashita, G. Ono, R. Nemoto, E. Suzuki, N. Masuda, T. Takemoto, F. Yuki, and T. Saito. A 12.3-mW 12.5-Gb/s complete transceiver in 65-nm CMOS Process. *IEEE Journal of Solid-State Circuits*, 45(12):2838–2849, 2010.

[15] E. Gabriel, G. E. Fagg, G. Bosilca, T. Angskun, J. J. Dongarra, J. M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine, et al. Open MPI: Goals, concept, and design of a next generation MPI implementation. In *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, pages 97–104. Springer, 2004.

[16] A. Gara. Energy Efficiency Challenges for Exascale Computing. In *ACM/IEEE Conference on Supercomputing: Workshop on Power Efficiency and the Path to Exascale Computing*, 2008.

[17] M. Ghosh and H.-H. S. Lee. Smart Refresh: An Enhanced Memory Controller Design for Reducing Energy in Conventional and 3D Die-Stacked DRAMs. In *Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 134–145, 2007.

[18] S. Huang, S. Xiao, and W. Feng. On the Energy Efficiency of Graphics Processing Units for Scientific Computing. In *IEEE International Symposium on Parallel and Distributed Processing (IPDPS)*, pages 1–8, 2009.

[19] A. A. Hwang, I. A. Stefanovici, and B. Schroeder. Cosmic Rays Don't Strike Twice: Understanding the Nature of DRAM Errors and the Implications for System Design. In *Proceedings of the 17th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 111–122, 2012.

[20] H. Kaul, M. Anders, S. Hsu, A. Agarwal, R. Krishnamurthy, and S. Borkar. Near-Threshold Voltage (NTV) Design: Opportunities and Challenges. In *Proceedings of the 49th ACM Annual Design Automation Conference*, pages 1153–1158, 2012.

[21] T. Kgil, A. Saidi, N. Binkert, S. Reinhardt, K. Flautner, and T. Mudge. PicoServer: Using 3D Stacking Technology To Build Energy Efficient Servers. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 4(4):16, 2008.

[22] J. Kim, N. Hardavellas, K. Mai, B. Falsafi, and J. Hoe. Multi-bit error tolerant caches using two-dimensional error coding. In *Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 197–209, 2007.

[23] J.-S. Kim, C. S. Oh, H. Lee, D. Lee, H.-R. Hwang, S. Hwang, B. Na, J. Moon, J.-G. Kim, H. Park, et al. A 1.2 V 12.8 GB/s 2Gb mobile Wide-I/O DRAM with 4× 128 I/Os using TSV-based stacking. In *IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pages 496–498, 2011.

[24] Y. Kim, V. Seshadri, D. Lee, J. Liu, and O. Mutlu. A case for exploiting subarray-level parallelism (SALP) in DRAM. In *39th IEEE Annual International Symposium on Computer Architecture (ISCA)*, pages 368–379, 2012.

[25] P. Kogge, K. Bergman, S. Borkar, D. Campbell, W. Carson, W. Dally, M. Denneau, P. Franzon, W. Harrod, K. Hill, et al. ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems. 2008.

[26] M. Kondo. Report on Exascale Architecture Roadmap in Japan, IESP Meeting, 2012.

[27] D. Lee, Y. Kim, V. Seshadri, J. Liu, L. Subramanian, and O. Mutlu. Tiered-Latency DRAM: A Low Latency and Low Cost DRAM Architecture. In *Proceedings of the 19th IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2013.

[28] S. Li, K. Chen, M.-Y. Hsieh, N. Muralimanohar, C. D. Kersey, J. B. Brockman, A. F. Rodrigues, and N. P. Jouppi. System Implications of Memory Reliability in Exascale Computing. In *IEEE International*

Conference for High Performance Computing, Networking, Storage and Analysis (SC), pages 1–12, 2011.

[29] X. Li, M. C. Huang, K. Shen, and L. Chu. An empirical study of memory hardware errors in a server farm. In *The 3rd Workshop on Hot Topics in System Dependability (HotDep'07)*, 2007.

[30] J. Liu, B. Jaiyen, R. Veras, and O. Mutlu. RAIDR: Retention-Aware Intelligent DRAM Refresh. In *39th IEEE Annual International Symposium on Computer Architecture (ISCA)*, pages 1–12, 2012.

[31] G. H. Loh. 3D-Stacked Memory Architectures for Multi-Core Processors. In *ACM SIGARCH Computer Architecture News*, volume 36, pages 453–464, 2008.

[32] C.-K. Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V. J. Reddi, and K. Hazelwood. Pin: building customized program analysis tools with dynamic instrumentation. In *ACM SIGPLAN Notices*, volume 40, pages 190–200, 2005.

[33] Micron. Technical Note TN-41-01: Calculating memory system power for DDR3, 2007.

[34] S. S. Mukherjee, J. Emer, T. Fossum, and S. K. Reinhardt. Cache scrubbing in microprocessors: Myth or necessity? In *Proceedings of the 10th IEEE Pacific Rim International Symposium on Dependable Computing*, pages 37–42, 2004.

[35] J. T. Pawlowski. Hybrid Memory Cube: Breakthrough DRAM Performance with a Fundamentally Re-Architected DRAM Subsystem. In *Proceedings of the 23rd Hot Chips Symposium*, 2011.

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

[37] V. Sarkar, S. Amarasinghe, D. Campbell, et al. Exascale Software Study: Software Challenges in Extreme Scale Systems. *DARPA Information Processing Techniques Office, Washington DC*, 14:159, 2009.

[38] B. Schroeder, E. Pinheiro, and W.-D. Weber. DRAM Errors in the Wild: A Large-Scale Field Study. In *Proceedings of the 11th ACM International Joint Conference on Measurements and Modeling of Computer Systems*, pages 193–204, 2009.

[39] V. Sridharan and D. Liberty. A Study of DRAM Failures in the Field. In *IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, pages 1–11, 2012.

[40] D. Tang, P. Carruthers, Z. Totari, and M. W. Shapiro. Assessment of the Effect of Memory Page Retirement on System RAS Against Hardware Faults. In *IEEE International Conference on Dependable Systems and Networks (DSN)*, pages 365–370, 2006.

[41] A. N. Udipi, N. Muralimanohar, N. Chatterjee, R. Balasubramonian, A. Davis, and N. P. Jouppi. Rethinking DRAM Design and Organization for Energy-Constrained Multi-Cores. In *ACM SIGARCH Computer Architecture News*, volume 38, pages 175–186, 2010.

[42] J. Vetter. On the Road to Exascale: Lessons from Contemporary Scalable GPU Systems. In *Proceedings of the ATIP/A* CRC Workshop on Accelerator Technologies for High-Performance Computing: Does Asia Lead the Way?*, page 27. A* STAR Computational Resource Centre, 2012.

[43] C. Weis, I. Loi, L. Benini, and N. Wehn. Exploration and Optimization of 3-D Integrated DRAM Subsystems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 32(4):597–610, 2013.

[44] D. H. Yoon and M. Erez. Virtualized ECC: Flexible Reliability in Main Memory. *IEEE Micro*, 31(1):11–19, 2011.

[45] H. Zheng, J. Lin, Z. Zhang, E. Gorbatov, H. David, and Z. Zhu. Mini-Rank: Adaptive DRAM Architecture for Improving Memory Power Efficiency. In *41st IEEE/ACM International Symposium on Microarchitecture*, pages 210–221, 2008.