Center for Embedded Computer Systems
University of California, Irvine

Towards Embedded RAIDs-on-Chip

Luis Angel D. Bathen and Nikil D. Dutt

Center for Embedded Computer Systems
University of California, Irvine
Irvine, CA 92697-2620, USA

{lbathen,dutt}@uci.edu

CECS Technical Report <10-12>
December 15, 2010

Towards Embedded RAIDs-on-Chip

LUIS ANGEL D. BATHEN, University of California, Irvine
NIKIL D. DUTT, University of California, Irvine

The dual effects of larger die sizes and technology scaling, combined with aggressive voltage scaling for power reduction, increase the error rates for on-chip memories. Traditional on-chip memory reliability techniques (e.g., ECC) incur significant power and performance overheads. In this paper, we propose a low-power-and-performance-overhead Embedded RAID (E-RAID) strategy and present Embedded RAIDs-on-Chip (E-RoC), a distributed dynamically managed reliable memory subsystem. E-RoC achieves reliability through redundancy by optimizing RAID-like policies tuned for on-chip distributed memories. We achieve on-chip reliability of memories through the use of Distributed Dynamic ScratchPad Allocatable Memories (DSPAMs) and their allocation policies. We exploit aggressive voltage scaling to reduce power consumption overheads due to parallel DSPAM accesses, and rely on the E-RoC manager to automatically handle any resulting voltage-scaling-induced errors. We demonstrate how E-RAIDs can further enhance the fault tolerance of traditional memory reliability approaches by designing E-RAID levels that exploit ECC. Finally, we show the power and flexibility of the E-RoC concept by showing the benefits of having a heterogeneous E-RAID levels that fit each application's needs (fault tolerance, power/energy, performance).

Our experimental results on multimedia benchmarks show that E-RoC's fully distributed redundant reliable memory subsystem can reduce up to 85% in dynamic power consumption, and up to 61% lower latency due to error checks/corrections. On average, we see that our E-RAID levels converge to 100% Yield much faster than traditional ECC approaches. Moreover, E-RAID levels that exploit ECC (e.g., E-RAID ECC + 1, E-RAID RP + ECC) can guarantee 99.9% Yield at ultra low Vdd on average, where as SECDED and DECTED were able to attain 99.1% and 99.4% Yield respectively. Our E-RAID levels (detection and correction) achieved a worst case 93.9% Yield, where as the traditional ECC approaches achieved a worst case of 34.1% Yield. We observe an average of 22% dynamic power consumption increase by using traditional ECC approaches (EDC1, EDC8, SEC, SECDED, DEC, DECTED), where as we observe average savings of 27% for our E-RAID schemes (E-RAID 1, E-RAID 1 + ECC, E-RAID ECC + 1, E-RAID RP, E-RAID RP + ECC, E-RAID TMR). We see that on average traditional ECC approaches are able to save static energy by 6.4%, where as our E-RAID approaches achieve 23.4% static energy savings. We observe that on average our approaches (E-RAID 1, E-RAID 1 + ECC, E-RAID ECC + 1, E-RAID RP, E-RAID RP + ECC, E-RAID TMR) incur 2% higher overheads than traditional ECC approaches (EDC1, EDC8, SEC, SECDED, DEC, DECTED). We observe that for Vdd above 0.45, on average, our E-RAID levels with error correction support (SEC) incur 3% lower overheads over the more traditional SECDED/DECTED schemes. Finally, we observe that mixing E-RAID levels allows us to reduce the dynamic power consumption by up to 150% at the cost of an average 5% increase in execution time over traditional approaches.

Categories and Subject Descriptors: C.3 [**Special-purpose and Application-based systems**]: Real-time and embedded systems; D.4.6 [**Security and Protection**]: Access Controls; Security Kernels; B.3 [**Design Styles**]: Virtual Memory; D.4 [**Storage Management**]: Distributed memories

General Terms: Design, Management, Performance, Security

This research was partially supported by NSF Variability Expeditions Award CCF-1029783, and SFS/NSF Grant No. 0723955.
Authors' addresses: Luis Angel D. Bathen and Nikil Dutt, Center for Embedded Computer Systems, School of Information and Computer Science, University of California at Irvine, Irvine, CA 92697;

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

Additional Key Words and Phrases: information assurance; security; chip-multiprocessors; policy; scratch-pad memory; virtualization; embedded systems

1. INTRODUCTION

Embedded system designs need to satisfy multiple constraints including power, performance and reliability. Continued technology scaling and larger die sizes, coupled with the increasing amounts of on-chip memory, make memories highly vulnerable to the threat of soft-errors and process variation induced errors [Ruckerbauer and Georgakos 2007; Mastipuram and Wee 2004; Nassif 2001]. Aggressive voltage scaling for power reduction further increases the error rates of on-chip memories [Sasan et al. 2009a; Kurdahi et al. 2010]. Traditional memory reliability techniques utilize ECC, or ECC-duplication hybrids, and incur significant power and performance overheads. This problem is exacerbated by the emergence of on-chip distributed memory subsystems, as evidenced by the trend of chip multiprocessor systems (e.g., IBM Cell [IBM 2005], Intel's Single-chip Cloud Computer [Intel 2009], Teraflops Research [Intel 2007], and Tilera Tile-Gx [Tilera 2010]), where cores can talk to multiple on-chip memories using different access/coherency protocols and a variety of communication infrastructures (e.g., bus matrix, P2P, NoCs, etc.).

As technology scales, system failure rates due to radiation-induced transient errors continues to be a major concern for embedded system designers [Lee et al. 2006]. Memories are most vulnerable to soft-errors since the total area of the die is dominated by memory cells. This problem worsens for chip-multiprocessor platforms that have even larger amounts of on-chip memory. Moreover, to reduce power consumption, designers employ techniques such as aggressive voltage scaling, which exponentially increases the impact of process variation on memory cells [Sasan et al. 2009a]. Voltage scaling reduces the capacitance that keeps the charge in a single cell, therefore affecting its vulnerability to low energy alpha particles, or cosmic rays [Mastipuram and Wee 2004]. Process variation is random in nature as it depends on many factors such as environmental (temperature, voltage), physical (mask imperfections, wear-out mechanisms), and in-die physical variations (layout, gate dimension). As process technology reaches its limits, failures due to process variation are rapidly increasing [Makhzan et al. 2007; Sasan et al. 2009b; Nassif 2001]. The probability of failure in SRAM technology is exponentially proportional to the decrease in voltage. Unlike soft-errors, which are transient in nature, process variation induced errors are permanent. Although aggressive voltage scaling increases the rate of failures, power savings can still be achieved by designing fault tolerant systems [Djahromi et al. 2007]. Efforts in reliable memory systems have focused on the design of error correction based memories, where data accesses are guarded by ECC mechanisms [Vergos and Nikolos 1995; Papirla and Chakrabarti 2009; Ghosh et al. 2004; Kim 2006; Kim et al. 2007; Ramaswamy and Yalamanchili 2007], replication based mechanisms [Lucente et al. 1990; Zhang 2004; Zhang et al. 2003; Li et al. 2005], as well as process variation aware designs [Makhzan et al. 2007; Sasan et al. 2009b; 2009a]. Note that some of these techniques may combine two or more different schemes to guarantee reliability of the memory subsystem. At the system level, Redundant Array of Inexpensive Disks (RAID) systems [Patterson et al. 1988] have been very successful in providing reliable data storage for the storage/distributed systems domain, and have been used from simple low cost servers to large scale storage area networks [Morris and Truskowski 2003], including operating environments that need to guarantee 24/7 uptime under heavy I/O loads.

This paper makes several contributions. Since distributed on-chip memory hierarchies are becoming common in chip-multiprocessor systems, we adapt and tune the traditional notion of RAID to define Embedded RAID (E-RAID) and Embedded RAIDs-on-Chip (E-RoC), a distributed dynamically managed reliable memory subsystem. Among

the key concepts introduced are: the notion of reliability via redundancy using an E-RAID system; a set of E-RAID levels that are optimized for use in embedded SoCs; the concept of distributed dynamic scratch pad allocatable memories (DSPAMs) and their allocation policies. We exploit aggressive voltage scaling to reduce power consumption overheads due to parallel DSPAM accesses. The resulting voltage-scale-induced errors that appear in the memories are handled by the E-RAID policies. We present the first proof-of-concept E-RoC Manager that exploits these ideas for Chip-Multiprocessors. We explore the flexibility and benefits of Embedded RAIDs-on-Chip by 1) studying their ability to complement existing fault tolerant approaches (e.g., ECC), 2) their ability to create a heterogeneous E-RAID level environment to match the different fault tolerance needs of each application, 3) the effects of arbitration policies, and 4) the power consumption/energy and performance overheads of various E-RAID levels.

2. BACKGROUND AND MOTIVATION

2.1. Background

Because of process variations, aggressive power saving techniques, technology scaling, hazardous environments, there are two major error types that threaten the integrity of data: transient soft-errors and permanent process variation induced errors. These types of errors have served as a motivation for many different techniques. ECC-based techniques [Vergos and Nikolos 1995; Papirla and Chakrabarti 2009; Ghosh et al. 2004; Kim 2006; Kim et al. 2007; Ramaswamy and Yalamanchili 2007] are both power and performance inefficient as the error checking/correction relies heavily on parity generation on each transaction. Duplication techniques focus mostly on cache based systems, and some even propose a secondary cache to keep track of duplicates [Zhang 2004; Zhang et al. 2003]. Software based schemes [Li et al. 2005] rely on the compiler to fully dictate how the blocks of data are mapped onto a single SPM. This motivates E-RoC, a power/performance/constraint-aware reliable memory system that exploits the idea of aggressive voltage scaling to reduce power consumption, and reduces the performance overhead inherent in reliable memory systems by exploiting the idea redundancy to validate data. The next two sections will go over the types of errors we are addressing.

2.2. Soft-Errors

Soft errors, i.e., transient faults, or single-event upsets (SEU), are caused primarily by external radiations in microelectronic circuits, and have been investigated extensively since the late 1970's [Lee et al. 2006].

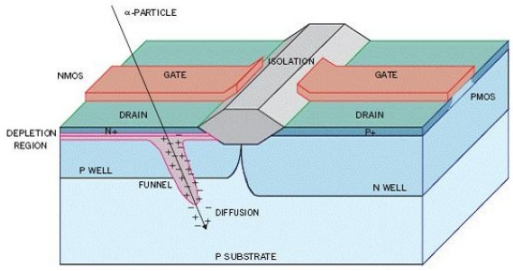

Fig. 1. Soft Error Event Occurrence in CMOS a Device [Lee et al. 2006].

Figure 1 shows a single soft-error event. As alpha or cosmic particles come into contact with the silicon device, if the charge of the given cell is low enough, its chances of suffering a single event failure (SEU) are lower. Because of techniques such as voltage scaling, the capacitance that keeps the charge in a single cell is reduced, therefore affecting its vulnerability to low energy alpha particles, or cosmic rays [Mastipuram and Wee 2004].

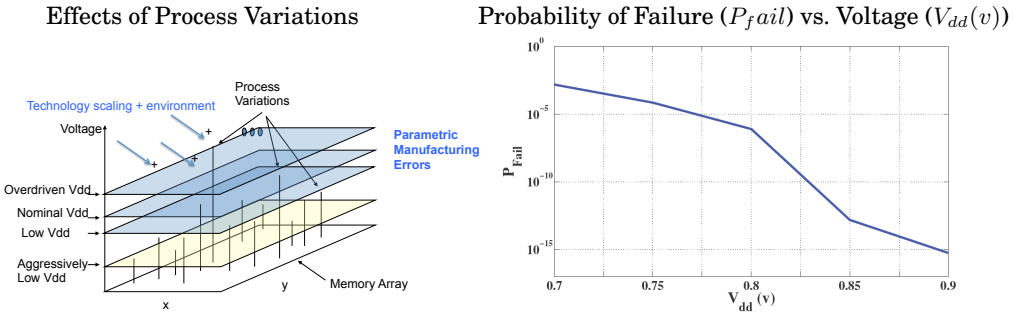

Fig. 2. Effects of Voltage Scaling [Kurdahi et al. 2010] (right) and Probability of Failure due to Voltage Scaling [Sasan et al. 2009a] (left).

2.3. Process Variation Errors via Voltage Scaling

Figure 1 (right) shows the effects of voltage scaling on a memory array. On the x and y-axis we have the memory array, and on the z-axis we have the voltage. The vertical lines show parametric manufacturing errors (process variations). In this figure we see that at overdriven Vdd, these process variations do not manifest on the memory array, but as we start lowering the voltage (e.g., Nominal Vdd, Low Vdd, Aggressively Low Vdd), we observe the process variations increasingly manifest on the memory array plane. This is further asserted by Figure 1 (left), which shows the effects of voltage scaling on the probability of failure for a single SRAM cell at 65nm technology. As we can observe, the probability of failure is exponentially proportional to the voltage (left) [Djahromi et al. 2007] .

3. CUSTOMIZING EMBEDDED RAIDS (E-RAIDS)

3.1. Traditional RAID Levels

RAID systems have been widely deployed since they were proposed in the 80's. RAID systems offer a wide array of levels. Each RAID level can be used to fit each application's needs. This section will briefly go over the most popular RAID levels.

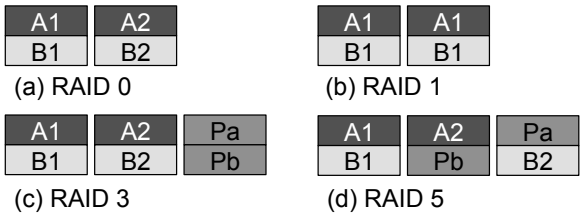

Fig. 3. Sample RAID Levels.

Figure 3 (a) shows a RAID 0 system, referred to as a striped set, where data is split into stripes and each stripe is then mapped onto a physical disk. The idea here is that a single transaction can be served in parallel by accessing multiple disks. A RAID 0 system requires a minimum of two disks, and the total capacity of the available disk space is given by Equation 1 , where DSi represents the disk size of drive i. If a disk fails, the whole RAID fails. RAID 0 is ideal for systems with high IO requirements. If a request for block A comes through, and RAID 0 accesses blocks A1 and A2 to serve the request for A.

$$C = \#disks\ in\ set \times min(DS_0, DS_1, ..., DS_n) \tag{1}$$

Figure 3 (b) shows a RAID 1 system, also known as mirror system, where each disk in the system contains a copy of the data. Like RAID 0, the disks are accessed in parallel. A minimum of two disks are needed. In case a disk fails, the second disk will continue to serve requests, thereby keeping the system from failing. The total capacity of RAID 1 is given by Equation 2. RAID 1 is ideal for systems that require high levels of reliability. If a request for block A comes through, and RAID 1 accesses block A1, if the disk fails it will then access A2. In the case of writes, block A will be written to both A1 and A2.

$$C = (\#disks\ in\ set/2) \times min(DS0, DS1, ..., DSn) \tag{2}$$

Figure 3 (c) shows a RAID 3 system, also known as striped with parity, uses byte-level striping, where the parity is stored on a separate disk. RAID 3 does not support parallel reads/writes as blocks are striped across multiple disks, therefore, each request for a block of data will have to access all disks in the RAID. RAID 4, is similar to RAID 3, the main difference is that RAID 4 uses block-level parity, therefore, accesses to different blocks may be serviced in parallel. Both RAID 3 and RAID 4 require at least three disks to build the RAID system. In the event of a single disk failure, the parity disk data may be used to reconstruct the lost data. The total capacity for RAIDs 3 and 4 is given by Equation 3. The main bottleneck for this level is the parity disk as it is accessed on every transaction.

$$C = (\#disks\ in\ set - 1) \times min(DS0, DS1, , DSn) \tag{3}$$

Among the RAID levels, RAID 5 is the most popular as it provides both performance and reliability. Like RAID 3 and 4, it follows the idea of parity, which is used to reconstruct failed disks. RAID 5 removes the bottleneck of the dedicated parity disk present in its predecessors by distributing the parity across multiple disks. Like RAID 3 and 4, the capacity of RAID 5 is given by (E3). Parity computation for RAID 5 are as simple as XORing two blocks of data, for instance, on a write of block A, RAID 5 stripes A into A1 and A2, and computes the parity A1 XOR A2 = Pa. It then writes both blocks and the parity.

It is possible to create RAID hierarchies. One of the most popular hierarchical RAID solutions is RAID 0 + 1 (10), where data is striped into blocks, thus helping the performance, and mirror copies of the blocks are kept to help reliability. For a more detailed overview of the different RAID technologies please refer to [Patterson et al. 1988; Morris and Truskowski 2003].

3.2. The Case for Embedded RAID

The goal of a traditional RAID system in storage systems is to guarantee the uptime of the system. In case a disk goes bad, the remaining disks are used to 1) serve data

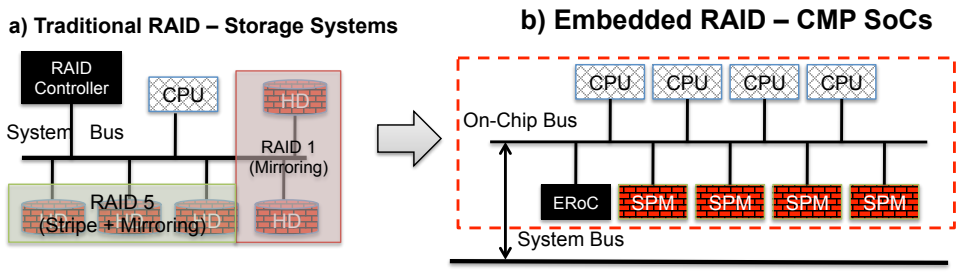

Fig. 4. Traditional RAID vs. Proposed Embedded RAIDs-on-Chip.

requests despite the failed disk, and 2) on disk replacement, rebuild the RAID system. In Embedded RAIDs (E-RAIDs), the notion of a failed SPM does exist; however, we cannot take the system offline, replace the SPM, and rebuild the E-RAID. Because of this, E-RAID levels need to be given a different purpose. The goal of an E-RAID is to guarantee the validity of the data stored in the E-RAID. Because of this, we must modify traditional RAID levels, and customize/optimize them for the use in embedded SoCs. Figure 4 (a) shows the traditional view of a RAID system, which consists of a CPU (set of CPUs), a set of $distributed$ hard drives configured in various RAID levels to meet the system's performance and fault tolerant needs (RAID 1, RAID 5, etc.), and a RAID hardware controller (could also have software RAID). Figure 4 (b) shows the proposed Embedded RAID model for Systems-on-Chip, which consists of a series of CPUs/Masters, the on-chip bus, a possible hardware manager (E-RoC Manager), and a set of $distributed$ on-chip memories (SPMs). As we can observe, these two system-level diagrams (though completely different abstractions) are quite similar.

3.2.1. RAID Requirement Overhead. One of the first arguments against RAIDs-on-Chip is the amount of extra memory space needed to keep a RAID system active. In the case of a RAID 1 system, the capacity is halved. In the case of RAID 1, 3-5 the capacity is reduced by at least a single SPM ($(\#SPMs-1)\times size\ of\ smallest\ SPM$). However, such overhead is comparable to the duplication techniques presented in [Lucente et al. 1990; Zhang 2004; Zhang et al. 2003; Angiolini et al. 2006; Li et al. 2005]. Moreover, due to process variation induced errors, available memory is reduced as data cells become unusable, and technology remapping techniques need to be used [Ramaswamy and Yalamanchili 2007]. Although resources are limited, research has shown that the memory subsystem is underutilized by current applications [Lucente et al. 1990; Zhang et al. 2003].

3.2.2. RAID Performance Overhead. Because the target devices are SPMs rather than disks, embedded RAID systems will benefit from parallel reads/writes to multiple memories. Of course, support for such model is needed. The major concern would be the parity calculation and checking. Since RAID system parity can be computed by a simple XOR, performance wise, embedded RAID systems offer a more performance friendly solution than any of the ECC/hybrid schemes previously proposed [Vergos and Nikolos 1995; Lucente et al. 1990; Papirla and Chakrabarti 2009; Zhang et al. 2003; Zhang 2004].

3.2.3. RAID Power Overhead. RAID systems may incur power overheads due to the extra accesses to memory systems necessary to access the duplicate data during reads and writes. However, in order to offset such power consumption, aggressive voltage scaling may be utilized, therefore efficiently reducing the penalty of the extra memory

accesses. Now, since each parity generation is done through XORs, the power consumption overhead per read/write is much lower than that of ECC/hybrid based schemes.

3.2.4. Which RAID Levels Make Sense. Because of the fact that both data and time scales are different for embedded systems and storage systems, we must be careful with the definition of RAID. The purpose of a RAID system in storage systems is to guarantee the uptime of the system. In the case a disk goes bad, the remaining disks are used to 1) serve data requests despite the failed disk, and 2) on disk replacement, rebuild the RAID system. In embedded RAIDs, the notion of failed SPM does exist; however, we cannot just simply replace the SPM, and rebuild' the RAID. One cannot take the system offline, replace the SPM, and rebuild the RAID. Because of this, RAID levels need to be given a different purpose. The goal of an embedded RAID is to guarantee the validity of the data stored in the RAID. Because of this, we must modify existing RAID levels, and optimize them for the use in the embedded system domain.

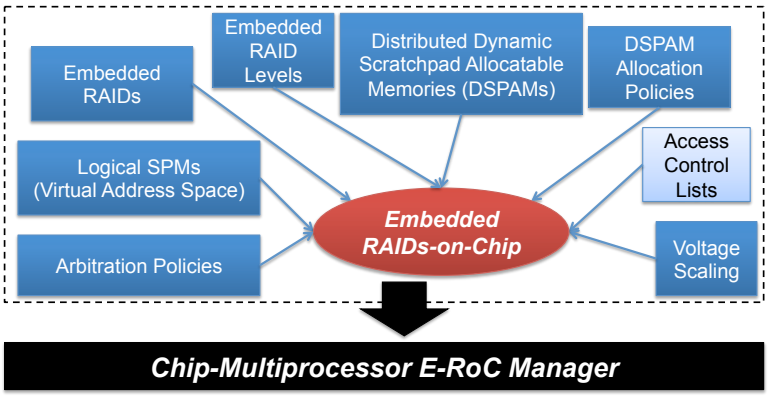

Fig. 5. Concept of Embedded RAIDs-on-Chip.

4. EMBEDDED RAIDS-ON-CHIP (E-ROC)

Figure 5 outlines the concept of Embedded RAIDs-on-Chip (E-RoC). *E-RoC* is composed of eight mutually dependent components that are used to create a customized Chip-Multiprocessor E-RoC Manager for the specific settings of each component. In this section, we will go over each of the different components of E-RoC, however, the main focus of this paper is to show the power of our E-RoC customization framework, thus we will focus more on the exploration and trade-off analysis (Yield, power/energy, performance) between the various E-RAID levels as well as their ability to complement built-in ECC schemes. In this paper, *we will focus primarily on Homogeneous Chip-Multiprocessor platforms* and will leave other platform configurations as future work (e.g., NoC-enabled Many-core Platforms).

4.1. Embedded RAIDs (E-RAIDs)

E-RAIDS exploit the idea that aggressive voltage scaling of memories significantly reduces power consumption, but increases the error rate in the memories. This intentional increase in the error rates can be automatically handled by E-RoC's RAID-like built-in error resiliency mechanisms. Thus in the E-RAID context, we use customized, reliable E-RAID levels to automatically handle the errors generated by aggressive

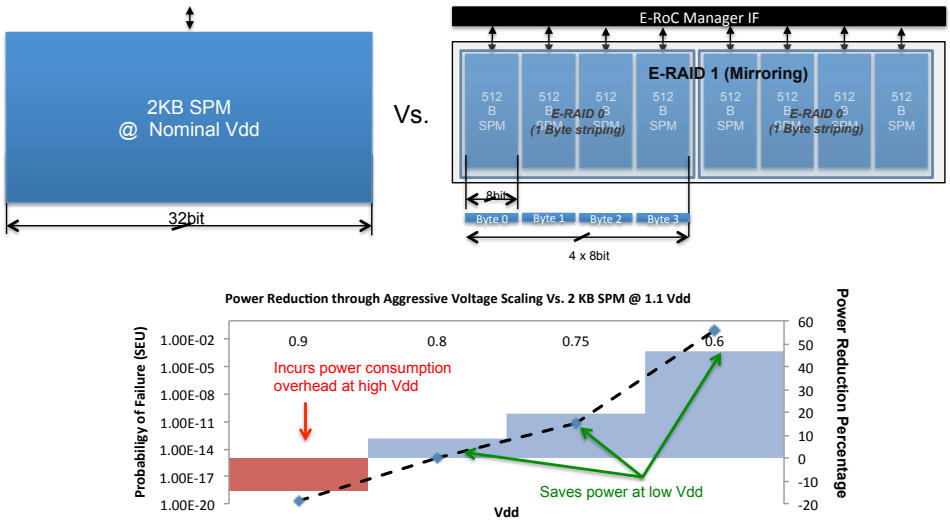

Fig. 6. Power Reduction for E-RAID 0+1 Level.

Table I. Average Read Response Time Vs. Read Size

Policy	ECC Mechanism	Redundancy Space	Notes
EDC1	1 bit parity	1 bit	Detection Only
EDC8	8 bit interleaved parity	8 bits	Detection Only
SEC	HAMMING	6 bits	Single Error Correction
SECDED	HAMMING	7 bits	Correction=1, Detection=2
DEC	CYCLIC	12 bits	Dual Error Correction
DECTED	CYCLIC	13 bits	Correction=2, Detection=3
E-RAID 1	NONE	32 bits	Detection only
E-RAID 1 + ECC	SEC	38 bits	SEC (Original) + Non-ECC Copy
E-RAID ECC + 1	SEC	38 bits	Non-ECC Copy + SEC (Original)
T-RAID 1	SECDEDx2	48 bits	Traditional RAID 1, ECC x 2 in parallel
E-RAID P	Parity	64 bits	Majority wins + Parity
E-RAID RP	Parity	32 bits	Keep 32 bit parity for detection
E-RAID RP + ECC	SEC	38 bits	Keep 32 bit parity and ECC for word
TMR	NONE	64 bits	Majority wins
NO E-RAID	NONE	0 bits	No E-RAID needed

voltage scaling of the memories. Moreover, as a side effect, transient errors are also automatically handled.

To illustrate the potential for power reduction using E-RAID refer to Figure 6, consider an E-RAID configuration consisting of eight 512B/8bit width SPMs voltage-scaled with data striping and replication (referred to as "E-RAID 0+1 Level"). Although the potential number of SPMs accessed per read/write transaction is 8 times that of a single SPM running at normal Vdd (1.1) and width of 32bits, *we observe up to 46% power reduction with aggressive voltage scaling*. Figure 6 shows progressive power savings due to aggressive voltage scaling in an E-RAID 0+1 configuration consisting of 65nm SPMs (gray bars). On the left axis we see that as voltage increases, the probability of failure increases exponentially as well (dotted line).

4.2. E-RAID Levels

Table I shows a series of traditional fault tolerant schemes (first half) and a subset of the supported E-RAID levels (second half) to illustrate how we have adapted traditional RAID levels to the on-chip context. Unlike existing duplication approaches [Pat-

terson et al. 1988; Angiolini et al. 2006], we do not allocate an entire memory to a single RAID level. We provide an API that allows for on-demand allocation/de-allocation of E-RAID space, which allows efficient use of the available on-chip resources. Each of the E-RAID levels provides a different degree of reliability guarantee proportional to its complexity and overheads. Our E-RAID levels allow for an extra on-chip memory refresh step in order to protect the memory subsystem against soft-errors, that is, if an error is detected and corrected, our E-RAID scheme writes back the corrected value. The refresh step is non-blocking, as the data is sent back to the masters/cpu and in parallel corrected in the memory system. Note that in the case of ECC-protected memories, the data banks are voltage scaled, and we assume that we have separate banks to store the ECC parities (G).

Algorithm 1 E-RAID Level 1 Read/Write Policy

Require: $REQ\_PACKET\{CTRL, ADDR, DATA^*\}$
1: **if** $REQ\_PACKET.CTRL == READ\_CTRL$ **then**
2: $A1 \leftarrow DSPAM_x(REQ\_PACKET.ADDR)$
3: $A2 \leftarrow DSPAM_y(REQ\_PACKET.ADDR)$
4: **if** $A1 == A2$ **then**
5: $REQ\_PACKET.DATA^* \leftarrow A1$
6: **return** $CHANNEL\_OK$
7: **else**
8: **return** SLV_ERR
9: **end if**
10: **else**
11: **if** $REQ\_PACKET.CTRL == WRITE\_CTRL$ **then**
12: $DSPAM_x(REQ\_PACKET.ADDR) \leftarrow REQ\_PACKET.DATA^*$
13: $DSPAM_y(REQ\_PACKET.ADDR) \leftarrow REQ\_PACKET.DATA^*$
14: **end if**
15: **end if**

4.2.1. E-RAID 1. E-RAID 1 follows the same redundancy idea of traditional RAID 1, also referred to as mirroring, where two copies of each block are kept in the E-RAID, each block being a 32bit word. As shown on Algorithm 1, on a read request, ***E-RoC*** fetches both copies of a data block, compares them, and returns the data if the comparison was successful (Lines 2-6). The methods $DSPAM_x()$ and $DSPAM_Y()$ perform the address translation for the transaction and are used to fetched/write/update the copies of the data in their respective memory regions (Lines 2-3). On an error the master will be forced to fetch the data from off-chip memory, thereby paying the penalty of a main memory access (Line 8). The master will then issue a write to this location to update the data (crucial in case of soft-errors). E-RAID 1 achieves lower power consumption and lower performance overheads than parity checking schemes as the comparison of the two blocks requires a simple *AND* or *XOR* and the reads/writes can be done in parallel. This approach assumes that the probability that two data blocks will have an error at the same location (bit) is very low [Zhang 2004; Zhang et al. 2003], however, unlike traditional replication approaches (e.g., [Zhang 2004; Zhang et al. 2003; Angiolini et al. 2006; Li et al. 2005]), E-RAIDs do not assume that the backup data is correct, thereby provide higher data-correctness guarantees.

4.2.2. E-RAID 1 + ECC. As shown in Algorithm 2, E-RAID 1 + ECC keeps two copies of the data, one backup and one original (protected by SEC). The idea is to minimize ECC overheads by comparing the two copies before the ECC check is done since a simple comparison incurs less overhead than the ECC check/correction. If the comparison fails, we try to correct it with SEC, if this fails, the data is fetched from off-chip memory just like E-RAID 1 (Lines 4-12). On a write (Lines 16-19), we perform the two writes

Algorithm 2 E-RAID Level 1 + ECC Read/Write Policy

Require: $REQ\_PACKET\{CTRL, ADDR, DATA^*\}$
1: **if** $REQ\_PACKET.CTRL == READ\_CTRL$ **then**
2: $A1 \leftarrow DSPAM_x(REQ\_PACKET.ADDR)$
3: $A2 \leftarrow DSPAM_y(REQ\_PACKET.ADDR)$
4: **if** $A1 == A2$ **then**
5: $REQ\_PACKET.DATA^* \leftarrow A1$
6: **return** $CHANNEL\_OK$
7: **else**
8: **if** $(TMP \leftarrow SEC(A1, DSPAM\_G_x(REQ\_PACKET.ADDR)))! = ERR$ **then**
9: $REQ\_PACKET.DATA^* \leftarrow TMP$
10: **return** $CHANNEL\_OK$
11: **else**
12: **return** SLV_ERR
13: **end if**
14: **end if**
15: **else**
16: **if** $REQ\_PACKET.CTRL == WRITE\_CTRL$ **then**
17: $DSPAM\_G_x(REQ\_PACKET.ADDR) \leftarrow SEC_G(REQ\_PACKET.DATA^*)$
18: $DSPAM_x(REQ\_PACKET.ADDR) \leftarrow REQ\_PACKET.DATA^*$
19: $DSPAM_y(REQ\_PACKET.ADDR) \leftarrow REQ\_PACKET.DATA^*$
20: **end if**
21: **end if**

to memory as in E-RAID 1, the only difference here is that we also perform the extra ECC step to protect the data element. This level requires us to have a small (6-bit) wide non-voltage scaled bank, which holds the parity.

Algorithm 3 E-RAID Level ECC + 1 Read/Write Policy

Require: $REQ\_PACKET\{CTRL, ADDR, DATA^*\}$
1: **if** $REQ\_PACKET.CTRL == READ\_CTRL$ **then**
2: $A1 \leftarrow DSPAM_x(REQ\_PACKET.ADDR)$
3: **if** $(TMP \leftarrow SEC(A1, DSPAM\_G_x(REQ\_PACKET.ADDR)))! = ERR$ **then**
4: $REQ\_PACKET.DATA^* \leftarrow TMP$
5: **return** $CHANNEL\_OK$
6: **else**
7: $A2 \leftarrow DSPAM_y(REQ\_PACKET.ADDR)$
8: **if** $(TMP \leftarrow SEC(A2, DSPAM\_G_y(REQ\_PACKET.ADDR)))! = ERR$ **then**
9: $REQ\_PACKET.DATA^* \leftarrow TMP$
10: **return** $CHANNEL\_OK$
11: **else**
12: **return** SLV_ERR
13: **end if**
14: **end if**
15: **else**
16: **if** $REQ\_PACKET.CTRL == WRITE\_CTRL$ **then**
17: $G \leftarrow SEC_G(REQ\_PACKET.DATA^*)$
18: $DSPAM_x(REQ\_PACKET.ADDR) \leftarrow REQ\_PACKET.DATA^*$
19: $DSPAM\_G_x(REQ\_PACKET.ADDR) \leftarrow G)$
20: $DSPAM_y(REQ\_PACKET.ADDR) \leftarrow REQ\_PACKET.DATA^*$
21: $DSPAM\_G_y(REQ\_PACKET.ADDR) \leftarrow G)$
22: **end if**
23: **end if**

4.2.3. E-RAID ECC + 1. Algorithm 3 shows the read/write policies for E-RAID ECC + 1, we first try to correct the data, if we are unable to do so, we fetch the second copy and try to correct it (Lines 2-14). The idea is to minimize the number of extra accesses incurred by E-RAID 1 and E-RAID 1 + ECC. As shown in Algorithm 3, this level incurs

the extra write needed to protect the copy (with respect to E-RAID 1 + ECC). Like E-RAID 1 + ECC, we assume that we can control the built-in ECC chip in the memory banks, which provides single error correction (SEC).

Algorithm 4 T-RAID Level 1 Read/Write Policy

Require: $REQ\_PACKET\{CTRL, ADDR, DATA^*\}$
1: **if** $REQ\_PACKET.CTRL == READ\_CTRL$ **then**
2: $ADDR \leftarrow REQ\_PACKET.ADDR$
3: $E1 \leftarrow SECDED(\&A1, DSPAM_x(ADDR), DSPAM\_G_x(ADDR))$
4: $E2 \leftarrow SECDED(\&A2, DSPAM_y(ADDR), DSPAM\_G_y(ADDR))$
5: **if** $E1! = ERR$ **then**
6: $REQ\_PACKET.DATA^* \leftarrow A1$
7: **return** $CHANNEL\_OK$
8: **else**
9: **if** $E2! = ERR$ **then**
10: $REQ\_PACKET.DATA^* \leftarrow A2$
11: **return** $CHANNEL\_OK$
12: **else**
13: **return** SLV_ERR
14: **end if**
15: **end if**
16: **else**
17: **if** $REQ\_PACKET.CTRL == WRITE\_CTRL$ **then**
18: $G \leftarrow SECDED_G(REQ\_PACKET.DATA^*)$
19: $DSPAM_x(REQ\_PACKET.ADDR) \leftarrow REQ\_PACKET.DATA^*$
20: $DSPAM\_G_x(REQ\_PACKET.ADDR) \leftarrow G)$
21: $DSPAM_y(REQ\_PACKET.ADDR) \leftarrow REQ\_PACKET.DATA^*$
22: $DSPAM\_G_y(REQ\_PACKET.ADDR) \leftarrow G)$
23: **end if**
24: **end if**

4.2.4. T-RAID 1. Algorithm 4 shows the read/write policies for the T-RAID level. T-RAID 1 executes SECDED x 2 in parallel and incurs both the extra access as well as the SECDED check overheads (Lines 2-4). If there is no error for either data block A1 or A2, then we can successfully return the data (Lines 5-11), otherwise, like E-RAID 1, the master will have to go off-chip and fetch the correct data element. This level provides the highest level of fault tolerance but may incur the highest overheads of all E-RAID levels as on every read and every write (Lines 17-23) T-RAID 1 must execute the SECDED checks/corrections.

Algorithm 5 E-RAID Level Random+Parity (RP) Read/Write Policy

Require: $REQ\_PACKET\{CTRL, ADDR, DATA^*\}$
1: **if** $REQ\_PACKET.CTRL == READ\_CTRL$ **then**
2: $A1 \leftarrow DSPAM_x(REQ\_PACKET.ADDR)$
3: $P \leftarrow DSPAM_p(REQ\_PACKET.ADDR)$
4: **if** $A1 \oplus R == P$ **then**
5: $REQ\_PACKET.DATA^* \leftarrow A1$
6: **return** $CHANNEL\_OK$
7: **else**
8: **return** SLV_ERR
9: **end if**
10: **else**
11: **if** $REQ\_PACKET.CTRL == WRITE\_CTRL$ **then**
12: $DSPAM_x(REQ\_PACKET.ADDR) \leftarrow REQ\_PACKET.DATA^*$
13: $DSPAM_y(REQ\_PACKET.ADDR) \leftarrow REQ\_PACKET.DATA^* \oplus R$
14: **end if**
15: **end if**

Algorithm 6 E-RAID Level Random+Parity (RP) Read/Write Policy - P Mapped to Non-Voltage Scaled Memory Space

Require: $REQ\_PACKET\{CTRL, ADDR, DATA^*\}$
1: **if** $REQ\_PACKET.CTRL == READ\_CTRL$ **then**
2: $A1 \leftarrow DSPAM_x(REQ\_PACKET.ADDR)$
3: $P \leftarrow DSPAM_p(REQ\_PACKET.ADDR)$
4: **if** $A1 \oplus R == P$ **then**
5: $REQ\_PACKET.DATA^* \leftarrow A1$
6: **return** $CHANNEL\_OK$
7: **else**
8: **return** $R \oplus P$
9: **end if**
10: **else**
11: **if** $REQ\_PACKET.CTRL == WRITE\_CTRL$ **then**
12: $DSPAM_x(REQ\_PACKET.ADDR) \leftarrow REQ\_PACKET.DATA^*$
13: $DSPAM_y(REQ\_PACKET.ADDR) \leftarrow REQ\_PACKET.DATA^* \oplus R$
14: **end if**
15: **end if**

4.2.5. E-RAID RP. The E-RAID RP schemes assume that the system has a large (32-bit) known prime number (R), and use it to construct the parity data (P) by XORing the data with R. This scheme is useful since R is a known value, so in case the data is corrupted, you can re-construct the value with R XOR P (assuming that P is stored in non-voltage scaled memory space). Note that the RP scheme is similar to the E-RAID 1 + P presented in [Bathen and Dutt 2011a], except that only a single copy is kept. E-RAID RP incurs the extra XOR overhead on every write to the E-RAID memory space. 5 shows the read/write policies for E-RAID RP assuming the parity (P) and the data block (A1) are stored in voltage scaled memory space, so the level is only useful for error detection and unlike E-RAID 1, it detects an error even if both A1 and P have a bit error in the exact same bit location. If P was stored in non-voltage scaled memory space, then on error detection (A1 has error), we could re-construct the data by XORing P and R as shown in Algorithm 6.

Algorithm 7 E-RAID Level Random+Parity (RP) Read/Write Policy + ECC (SEC)

Require: $REQ\_PACKET\{CTRL, ADDR, DATA^*\}$
1: **if** $REQ\_PACKET.CTRL == READ\_CTRL$ **then**
2: $A1 \leftarrow DSPAM_x(REQ\_PACKET.ADDR)$
3: $P \leftarrow DSPAM_p(REQ\_PACKET.ADDR)$
4: **if** $A1 \oplus R == P$ **then**
5: $REQ\_PACKET.DATA^* \leftarrow A1$
6: **return** $CHANNEL\_OK$
7: **else**
8: **if** $(TMP \leftarrow SEC(A1, DSPAM\_G_x(REQ\_PACKET.ADDR)))! = ERR$ **then**
9: $REQ\_PACKET.DATA^* \leftarrow TMP$
10: **return** $CHANNEL\_OK$
11: **else**
12: **return** SLV_ERR
13: **end if**
14: **end if**
15: **else**
16: **if** $REQ\_PACKET.CTRL == WRITE\_CTRL$ **then**
17: $DSPAM_x(REQ\_PACKET.ADDR) \leftarrow REQ\_PACKET.DATA^*$
18: $DSPAM_y(REQ\_PACKET.ADDR) \leftarrow REQ\_PACKET.DATA^* \oplus R$
19: $DSPAM\_G_x(REQ\_PACKET.ADDR) \leftarrow G)$
20: **end if**
21: **end if**

Algorithm 8 E-RAID Level Random+Parity (RP) Read/Write Policy + ECC (SEC - P Mapped to Non-Voltage Scaled Memory Space

Require: $REQ\_PACKET\{CTRL, ADDR, DATA^*\}$
1: **if** $REQ\_PACKET.CTRL == READ\_CTRL$ **then**
2: $A1 \leftarrow DSPAM_x(REQ\_PACKET.ADDR)$
3: $P \leftarrow DSPAM_p(REQ\_PACKET.ADDR)$
4: **if** $A1 \oplus R == P$ **then**
5: $REQ\_PACKET.DATA^* \leftarrow A1$
6: **return** $CHANNEL\_OK$
7: **else**
8: **if** $(TMP \leftarrow SEC(A1, DSPAM\_G_x(REQ\_PACKET.ADDR)))! = ERR$ **then**
9: $REQ\_PACKET.DATA^* \leftarrow TMP$
10: **return** $CHANNEL\_OK$
11: **else**
12: **if** $(TMP \leftarrow SEC(R \oplus P, DSPAM\_G_x(REQ\_PACKET.ADDR)))! = ERR$ **then**
13: $REQ\_PACKET.DATA^* \leftarrow TMP$
14: **return** $CHANNEL\_OK$
15: **else**
16: **return** SLV_ERR
17: **end if**
18: **end if**
19: **end if**
20: **else**
21: **if** $REQ\_PACKET.CTRL == WRITE\_CTRL$ **then**
22: $DSPAM_x(REQ\_PACKET.ADDR) \leftarrow REQ\_PACKET.DATA^*$
23: $DSPAM_y(REQ\_PACKET.ADDR) \leftarrow REQ\_PACKET.DATA^* \oplus R$
24: $DSPAM\_G_x(REQ\_PACKET.ADDR) \leftarrow G)$
25: **end if**
26: **end if**

4.2.6. E-RAID RP + ECC. E-RAID RP + ECC enhances with E-RAID RP by keeping the ECC for the data as backup in case the parity (P) is unable to reconstruct the data (Algorithm 7). The main difference here is that like E-RAID 1 + ECC, on an error detection, it attempts to correct it using SEC, if it is unable to correct the error, then the master/CPU will have to fetch the data from off-chip memory. Algorithm 8 shows the enhanced version of E-RAID RP + ECC, which assumes that the parity (P) is stored in non-voltage scaled memory, and can attempt to correct the data by using SEC on the result of $R \oplus P$.

4.2.7. E-RAID TMR. Triple Modular Redundancy (TMR) follows the same notion as E-RAID 1, but maintains three copies (instead of two), and uses a majority vote to generate the correct result. E-RAID levels provide a degree of reliability, from simple mirroring (E-RAID 1) to complex parity checking (E-RAID RP), however, not all data might need reliability guarantees.

4.2.8. NO E-RAID. The NO E-RAID level consists of voltage scaled logical DAMes and allows for raw access to DAMe space with no reliability guarantee. This level is extremely useful when the applications running on the CMP are error tolerant (e.g., Multimedia), where we can tolerate errors in non-critical (e.g., pixel) data at the expense of lower quality-of-service (QoS).

4.2.9. Mixing E-RAID Levels. E-RAIDs are flexible enough to allow a designer/compiler/OS to choose the right level for a given piece of data. Like stated in Section 4.2.8, we can combine various E-RAIDs depending on the application's needs. Our approach can exploit memory subsystems with built-in ECC and memory subsystems where ECC is not present. It is possible to voltage scale some memories (SPMs), and let the E-RoC Manager know which memories are voltage scaled, which memories have ECC support

(and ECC strength), and number of memories it has access to. Given these parameters, the programmer/compiler will make the decisions for the types of E-RAIDs needed to execute the given application, and the E-RoC Manager will handle to mapping from E-RAIDs to physical SPMs (DSPAMs).

4.3. Complementing Existing Fault Tolerant Schemes

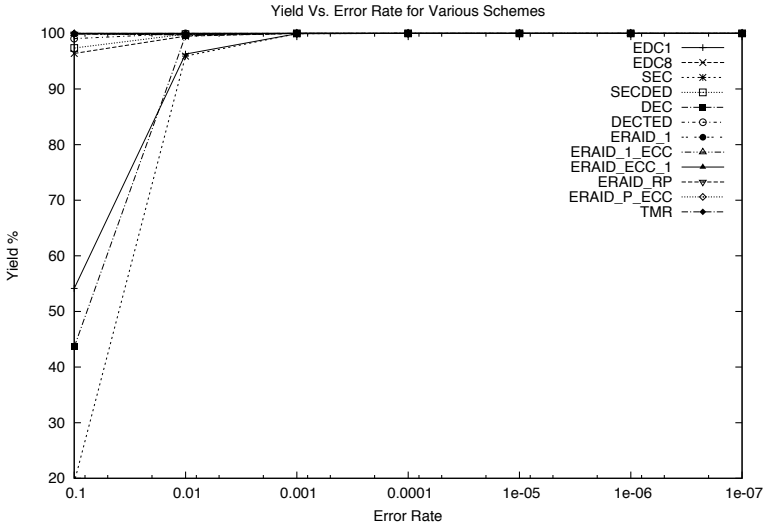

Fig. 7. Yield % vs. Error Rate for Various Schemes Assuming Backup Data in Main Memory.

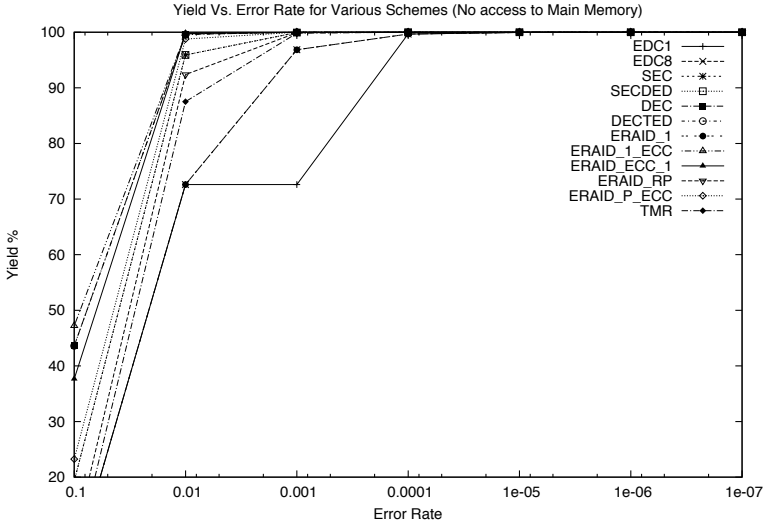

Fig. 8. Yield % vs. Error Rate for Various Schemes Assuming no Backup Data.

E-RAID levels can exploit and enhance existing ECC/Cyclic schemes to provide higher reliability with minimal overheads. Figure 7 shows various ECC/Cyclic error detection and correction schemes as well as a series of E-RAID levels and compares their Yield as a function of the error rate (#errors/bit cell) under different assumptions. From Figure 7 we can observe that our E-RAID (E-RAID+SEC) levels can further enhance the Yield of a system (up to 59% at high error rates ($>$0.1 errors/bit cell) delivers an average Yield improvement of 8.9% across all error rates. Note that the assumption here is that on error detection, an error-free backup data can be fetched from off-chip memory. Figure 8 shows the case where there is no backup data in off-chip memory, and thus the Yield rate is quite low for the schemes with error-detection only mechanism: EDC1, EDC8, [Kim et al. 2007] and ERAID_1. Even under the assumption that there is no backup data in off-chip memory, we can see that E-RAID levels can still provide better Yield than traditional approaches with error correction mechanisms (even the hardened and costly DECTED). As shown in Figure 5, E-RAID levels are one of the key components in *E-RoC* as different applications might have different fault-tolerance, power, and performance requirements. Our E-RoC customization framework allows designers to explore among the various levels and choose the one that meets their power/performance needs.

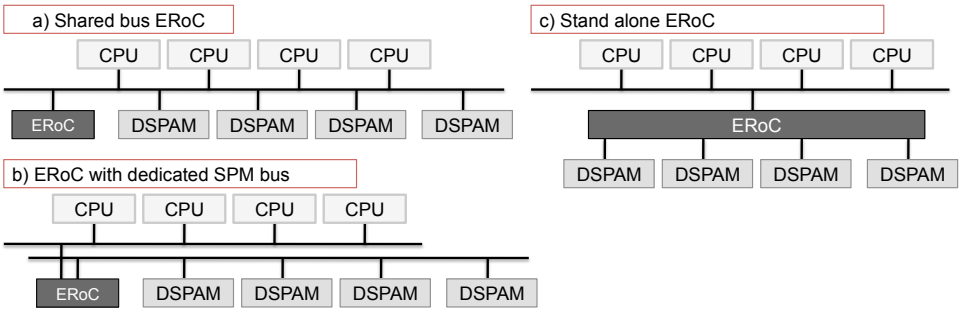

Fig. 9. Platform Configurations.

4.4. Multi-platform Support and Dynamic Scratch Pad Allocatable Memories (DSPAMs)

We target our designs for multiprocessor embedded SoCs where each processing core may need to configure an E-RAID system to handle its executing task. As shown in Figure 9, E-RoC can be customized for different architectural platforms. Since this is the first piece of work introducing E-RoC, our goal is to show the use of E-RoC on familiar platforms. Thus we consider a simple homogeneous CMP architecture, consisting of multiple processing cores (CPUs), instruction cache, distributed SPMs, a DMA engine to facilitate the data transfers among the various SPMs, and a shared bus topology.

We introduce the notion of distributed Dynamic Scratch Pad Allocatable Memories (DSPAMs). These memories differ from SPMs in that although they are still part of the memory space, they are only accessed by/through the E-RoC module, and are aggressively voltage scaled. Their physical space is dynamically allocated/de-allocated by the E-RoC module using a variety of platform configurations. For instance, Figure 9 (a) shows a platform configuration consisting of an E-RoC module connected to the on-chip bus, with the E-RoC module responsible for maintaining E-RAID systems for each CPU. Each E-RAID data request will be routed by the bus to the E-RoC slave, which in turn sends slave requests to each of the respective E-RAID memories it manages. This model suffers from delays due to on-chip bus traffic. The benefits of a shared model

however, are the flexibility in managing the available DSPAMs. Figure 9 (b) shows a second configuration which consists of a dedicated DSPAM bus where each E-RAID request is routed to the E-RoC module, which then issues read requests to each of the DSPAMs. Unlike the model in Figure 9 (a), this model does not suffer from delays due to the extra traffic on the main on-chip bus. Figure 9 (c) shows a third configuration, which consists of a single stand-alone E-RoC module that has point-to-point connectivity to each of its managed DSPAMs. This point-to-point connectivity further reduces the delay due to on-chip data transfers, since each E-RoC request to each of its managed DSPAMs can be processed instantaneously. One drawback from this model is that the available DSPAM resources are pre-defined, and therefore not as flexible as the previous two models.

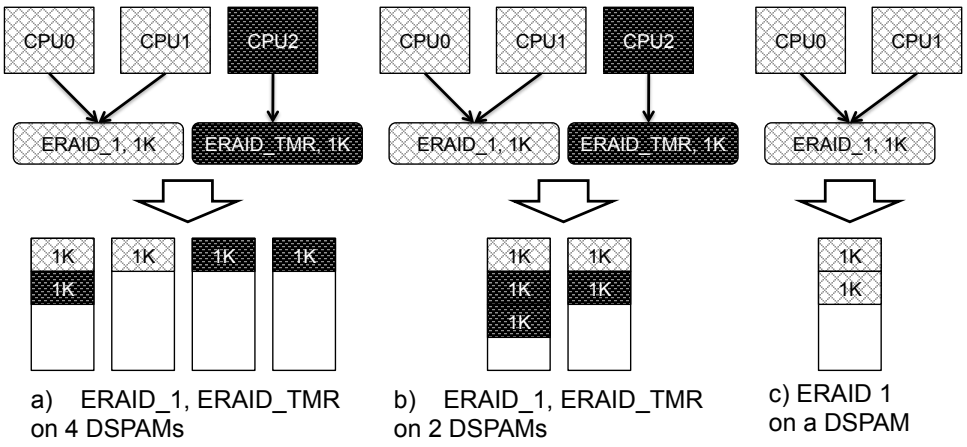

Fig. 10. DSPAM Allocation Policies.

4.5. DSPAM Allocation Policies

Since DSPAMs are extremely valuable resources, the DSPAM allocation algorithm must be optimized to efficiently allocate their space for each E-RAID system. In traditional storage systems, an entire disk is dedicated to a RAID system. If we were to fully allocate a DSPAM to an E-RAID system, the performance degradation would be too large. To account for this overhead, we introduce two ideas: (i) a virtual DSPAM address space, which is a unified global address space viewed by the external world, and (ii) the dynamic allocation of this virtualized address space. The virtual address space makes it easy for the compiler to allocate/de-allocate space as it would if it was targeting a regular SPM, and the dynamic support enables us to configure E-RAID systems of various sizes at run-time, thereby providing the necessary memory space for each task.

Figure 10 shows three different examples where block based allocation successfully configured E-RAID systems for two different processors. CPU0 requested a 1K E-RAID 1 system, which will be shared with CPU1, and CPU2 requested an E-RAID TMR 1K system. The first two designs with (a) four and (b) two 4KB DSPAMs respectively successfully created the E-RAID systems. The third design (c) with a single 4KB DSPAM has successfully allocated space for the E-RAID 1 1K system, and returned a SLV_ERR for the allocation of the second E-RAID TMR system request since there are no more resources available for its creation. The first allocation policy we

Algorithm 9 $AllocationPolicy(v\_start\_addr, v\_end\_addr, DSPAM_{num}, E-RAID_{size})$

Require: $v\_start\_addr, v\_end\_addr, DSPAM_{num}, E-RAID_{size}$
1: {input: virtual start/end addresses, number of DSPAMs to use, and the desired E-RAID size}
2: {output: an E-RAID ID - 0 if unable to allocate space}
3: $tries \leftarrow DSPAM_{dedicated}$
4: $DSPAM_{cnt} \leftarrow DSPAM_{num}$
5: $entries\_created \leftarrow 0$
6: **while** $DSPAM_{cnt}>0 \wedge tries>0$ **do**
7: $found\_block \leftarrow false$
8: $block \leftarrow 0$
9: **for** $j \leftarrow 0; j<ROC\_BLOCK\_SIZE \wedge !found\_block; j++$ **do**
10: **if** $MAP[rr\_DSPAM\_malloc\%DSPAM_{dedicated}][j] == 0$ **then**
11: $block++$
12: **else**
13: $block \leftarrow 0$
14: **end if**
15: **if** $(block-1) \times ROC\_BLOCK\_SIZE == E-RAID_{size}$ **then**
16: $found\_block \leftarrow true$
17: $p\_addr\_s \leftarrow (j-(block-1)) \times ROC\_BLOCK\_SIZE$
18: $p\_addr\_e \leftarrow j \times ROC\_BLOCK\_SIZE$
19: {update spm blocks occupied}
20: **for** $i \leftarrow (j-(block-1)); i \leq j; i++$ **do**
21: $MAP[rr\_DSPAM\_malloc\%DSPAM_{dedicated}][j] = 1$
22: **end for**
23: {create entry}
24: $create\_lut\_entry(v\_start\_addr, v\_end\_addr, p\_addr\_s, p\_addr\_e)$
25: $entries\_created++$
26: $DSPAM_{cnt}--$
27: **end if**
28: **end for**
29: $rr\_DSPAM\_malloc++$
30: $tries--$
31: **end while**
32: **if** $entries\_created == DSPAM_{num}$ **then**
33: **return** $E-RAID\_ID++$
34: **end if**
35: **return** 0

explore follows the Next Fit model, where DSPAM memory space is split into k blocks ($k \in 64, 128, 256, 512\ bytes$). For each DSPAM we keep a free list (single bit), and we allocate on the next best-fit basis, while maintaining a circular list of free blocks per DSPAM. The number of SPMs searched in parallel for allocation depends on the E-RAID level (e.g. E-RAID 1 requires 2 DSPAMs). We walk through DSPAMs in a round-robin mode in order to fairly distribute data across DSPAMs. Like traditional RAID single disk failures, unusable DSPAMs are removed from the list of managed DSPAMs and background E-RAID re-mapping is done.

Algorithm 9 shows our allocation algorithm which uses a round robin first fit block allocation scheme. The algorithm walks through each available DSPAM (SPM dedicated to E-RoC) in round robin fashion in order to evenly distribute data blocks among the different DSPAMs. For each DSPAM, if it encounters an available data block (Line 10). It adds the block to the list. If the size of the block list matches the desired E-RAID system size (line 15), it marks all blocks in the list as occupied (Lines 20-22). At the same time, the physical DSPAM start and end address are computed (Lines 17, 18) and a LUT entry is created for the physical/virtual address pairs (Line 24). If it does not find such list, then it moves to the next DSPAM until the number of tries reaches zero (a full round robin cycle). If the E-RAID system is successfully created, the allocation method will then return the E-RAID ID (Line 33). Otherwise it returns

0. Block availability information is kept in block MAPs, where a 0 in the $DSPAM_i$'s j^{th} entry means that the corresponding block in the DSPAM is free, a 1 means occupied. The block size can be configured, currently, we support block sizes of 64Bytes, 128 Bytes, 256 Bytes and 512 Bytes. The back end contains a look up table that maps virtual DSPAM addresses which are the addresses controlled by the compiler to physical DSPAM addresses, allowing a fast address translation mechanism to process E-RAID requests. Each E-RAID system will contain a set of LUT entries which will allow it to map virtual addresses to physical DSPAM addresses. All information for an E-RAID system is stored in configuration memory. Because of this, the number of concurrent E-RAID systems supported is limited to one per master, at a maximum of 32 masters. However, if we increment the available memory in E-RoC from 1K to say 4K, we can quadruple this number. Note that we could potentially have fixed block mapping, which would allow for faster address translation and less fragmentation at the cost of extra E-RoC configuration memory space, as well as more complex allocation policies such as variable size block allocation [Francesco et al. 2004].

The de-allocation algorithm walks through the LUT entries corresponding to the E-RAID ID being deleted, and clears the MAP entries (sets them to 0) for each of them. This enables the Allocation algorithm to use these freed blocks to create new E-RAID systems.

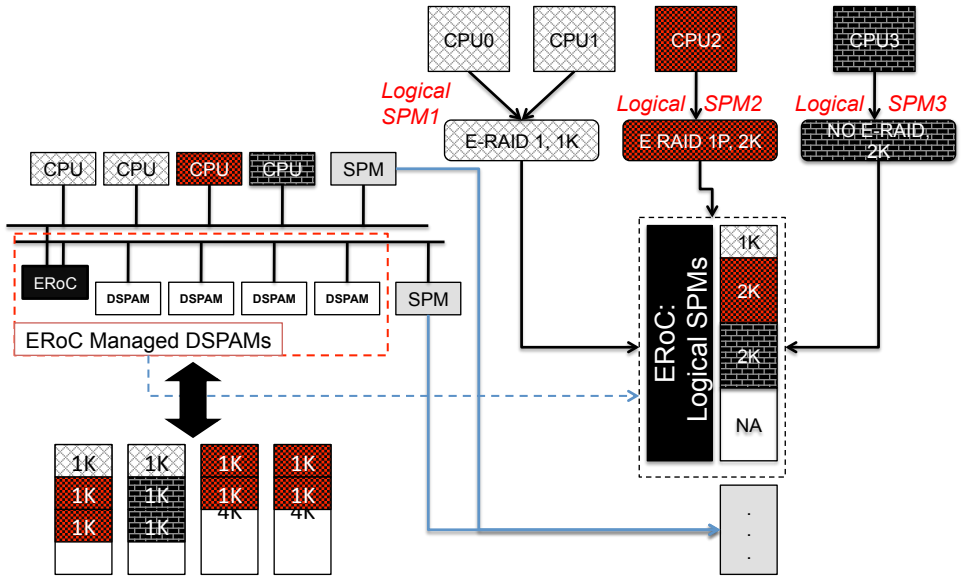

Fig. 11. Virtual Address Space.

4.6. Logical SPMs and Virtual Address Space

Because data placement onto the E-RAID systems is still left to the compiler, we must make it possible for the compiler to configure E-RAIDs on demand, as well as manage the data efficiently. We introduce the idea of Logical SPMs via virtual address spaces in E-RAID systems. One of the main benefits of E-RoC is that it abstracts the complexity of the E-RAID system, and presents the compiler with a simplified view of the address space via a logical SPM. The compiler can create an E-RAID system, and regardless of the E-RAID level being enforced, the compiler will see the E-RAID system as a

logical memory mapped DSPAM. All transfers between the CPU(s), main memory and the E-RAID system will follow the same process as in a system with pure SPMs and DMA support. As an example, Figure 11 shows a diagram of a 4 CPU CMP with SPM support, and an E-RoC manager with 4 DSPAMs. As shown, CPUs 0 and 1 configured a 1K E-RAID 1 shared system, CPU2 configured a 2K E-RAID 1 + P system, and CPU3 has no E-RAID, but wants to access the DSPAM space. All accesses to the E-RAID systems are transparent as the CPUs see their E-RAIDs as logical SPMs. The virtual address space presented to the outside world is shown in the dark dashed lined box. The main difference between DSPAMs and Logical SPMs is that DSPAMs are physical voltage scaled SPMs visible to the E-RoC manager, while Logical SPMs are addressable memory spaces visible to the compiler/OS. Note that a Logical SPM must $first$ be created before an E-RAID level can be associated with that Logical SPM, this is reflected in Figure 11, where Logical SPM 1 is associated with an E-RAID Level of 1KB.

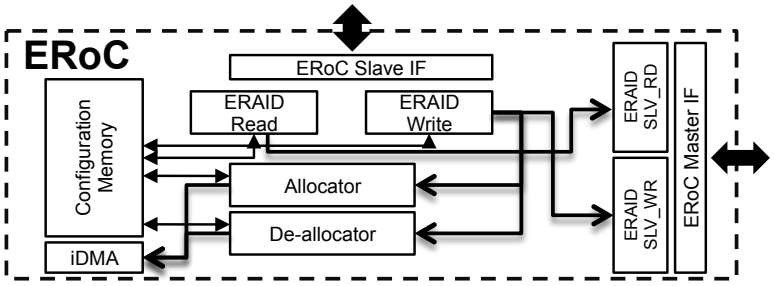

Fig. 12. E-RoC Manager Architecture.

4.7. E-RoC Manager Architecture

Figure 12 shows the architecture of the E-RoC manager, consisting of a master and a slave interface. The slave interface handles incoming E-RAID requests, and the master interface issues read/write requests to each DSPAM managed. Each master in the system has a dedicated and restricted memory space in the E-RoC module. This mechanism prevents other masters from overwriting configuration information for another master's E-RAID system. The E-RAID Read/Write modules handle read/write requests depending on the policies being handled. If the transaction is a configuration request, then the validity of the request is checked. Given the allocation policy, on an E-RAID create request, the allocator searches for space across the various DSPAMs; if there is enough space, the E-RAID system is created. In the case the master desires the E-RAID system to be built using data from some memory space, the allocator uses its internal DMA engine (iDMA) to fetch the data and store it in the new E-RAID system. On a de-allocation request, the E-RAID is offloaded onto main memory (when desired) and the blocks occupied by the E-RAID are freed.

Figure 13 shows the configuration packet for the E-RoC manager. Because we are dealing with software controlled memories (SPMs), what data is placed onto the E-RAID is still left to the compiler. Therefore each E-RAID system must be configured by a system master with the support of the compiler. E-RoC provides a set of API calls to allocate/de-allocate an E-RAID system. Each master may have at most one configured E-RAID, however, because of the fact that E-RoC supports access control lists (ACLs), each master may be able to access more than one E-RAID system at a time. At the core

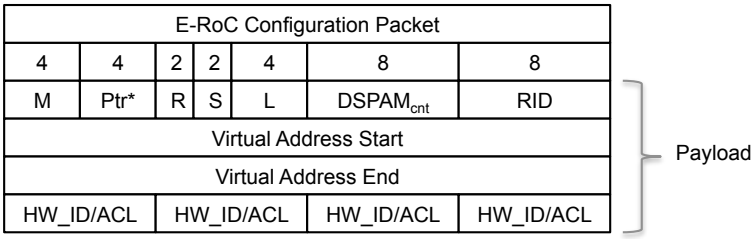

Fig. 13. E-RoC Manager Configuration Packet.

of the API is the configuration packet sent to the E-RoC. Every time an E-RAID is to be configured/created/deleted a configuration packet must be constructed and sent to the E-RoC manager. Each configuration packet is written to the configuration memory space dedicated to the CPU sending it to the RoC module. The configuration packet consists of the following fields:

— RID: The E-RAID ID populated by the Allocator block on a successful E-RAID creation. It is 0 otherwise
— $DSPAM_{cnt}$: Number of DSPAMs managed by the E-RAID level
— L: E-RAID level to be configured
— S: Field used to determine if an E-RAID is shared
— M: The mode field is used to create/delete E-RAID systems. There are four modes supported: CREATE, CREATE LOAD, DELETE, DELETE UNLOAD. Both CREATE and DELETE simply create/delete the E-RAID systems. CREATE LOAD and DELETE LOAD load/unload data from/to main memory via iDMA transactions
— Virtual Start and End Addresses refer to the virtual addresses as viewed by the program in SPM space
— ACLs: Each E-RAID system may be configured by a single master. No other master can access the E-RAID's data. A master may grant access to its E-RAID to other masters as long as their respective IDs are added to the ACL (at most 5 masters may share a Logical SPM (and an E-RAID Level)
— Ptr^*: A pointer/address field which is used to associate a level with a Logical SPM

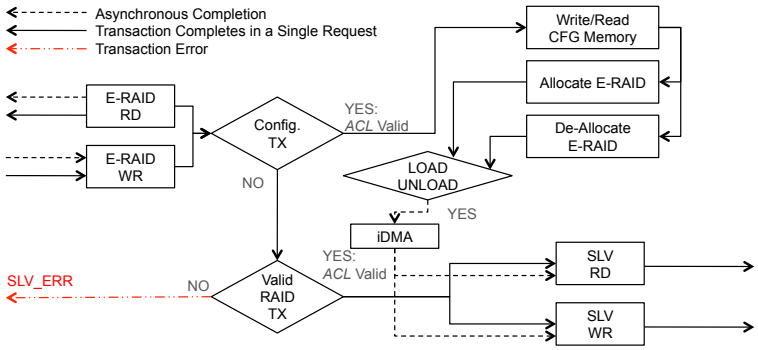

Fig. 14. E-RoC Manager Configuration Packet.

Figure 14 shows the control sequence for carrying transactions by the E-RoC manager. If a transaction is deemed as a valid configuration memory transaction, the

packet is written to CFG memory, in case of a read request, the configuration memory is read and returned to the master. On a write, the mode is checked. If it is a create request, the allocator is invoked and an E-RAID is created, in case it is not feasible (not enough space), then an error (SLV_ERR) is returned to the master. In case the E-RAID was successfully created, then the mode is checked for the LOAD option, if so, then a DMA request is places for the data to be loaded onto the E-RAID via iDMA. Similarly, on a delete UNLOAD option, the iDMA is configure to transfer the data from the E-RAID onto main memory. At this point, the request returns to the master, however, it is an asynchronous completion and thus the master has to wait for its E-RAID data to be loaded/unloaded (this process is shown by the dashed dark edges) before it use or can create a new E-RAID system. On a simple create/delete request or E-RAID valid transaction (TX), the transaction completes and is sent back to the master (shown by the dark straight edges). Note that before accessing any memory region, the transaction's ACL is validated.

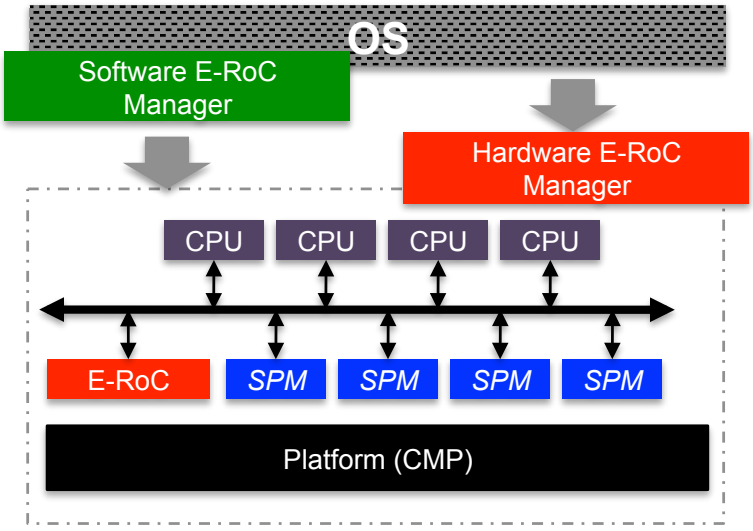

Fig. 15. Software/Hardware E-RoC Layer.

4.8. Software/Hardware E-RoC Layer

As discussed in section 3.2, like traditional RAID, the E-RoC concepts can be implemented as a Hardware module (e.g., an enhanced bus arbiter with MMU and internal DMA capabilities as shown in the previous section) or a Software memory management module (like software RAID) as shown in Figure 15. The software E-RoC layer should be light-weight, flexible, and modularized in a manner that allows for easy integration into existing OSes/Hypervisors. The hardware E-RoC Manager module should have minimal area overheads, and support a simplified API for transparent use by the programmers or the OS/Hypervisor software stacks. The software E-RoC Manager Module has the benefit of being flexible, portable (across various hardware configurations) and requires no extra hardware. The benefits of the software implementation comes at the cost of higher power/performance overheads than the hardware implementation. Ideally, both hardware and software E-RoC should support the same minimal API and should require minimal changes in the programming model. For this pa-

per, we mainly focused on the hardware E-RoC implementation, and leave the software E-RoC implementation as future work. Our goal is to have a tightly coupled SW/HW layer that exploits the benefits of both software and hardware E-RoC modules.

4.9. Access Control Lists (ACL)

Access Control Lists (ACLs) are used to guarantee that no unauthorized masters gain access to a given memory region (E-RAID), this is to protect memory regions from accidental and malicious accesses to memory regions, thereby protecting against data corruption and possible eavesdropping by malicious processes [Bathen and Dutt 2011b].

5. RELATED WORK

SPMs have through the years become a critical component of the memory hierarchy [Jung et al. 2010; Bai and Shrivastava 2010], and are expected to be the memories of choice for future many-core platforms (e.g., [IBM 2005; Intel 2009; Tilera 2010]). Unlike cache-based platforms where data is dynamically loaded into the cache with hopes of some degree of reuse due to access locality, SPM based systems depend completely on the compiler to determine what data to load. Placement of data onto memory is often done statically by the compiler through static analysis or application profiling, the location of data is known a priori which increases the predictability of the system [Panda et al. 1997] . For these reasons we focus on exploiting distributed SPMs.

Much of the state of the art work in reliable memory systems focuses on caches. [Makhzan et al. 2007] propose the idea of exploiting error maps to correct faulty cells on the main cache. [Kim 2006] introduce an area efficient ECC cache protection mechanism. [Kim et al. 2007] proposed a multi-bit error correction scheme for Caches using 2D-ECC. [Lee et al. 2006] propose the idea of using partitioned caches to protect critical data that is mapped onto an ECC protected cache, with non-critical data mapped onto a regular cache. [Zhang 2004] introduce a small fully associative cache into the memory hierarchy where data is replicated; the duplicates are used to detect and correct errors. In the event of process variation errors, techniques such as technology mapping and cache redundancy are used [Lucente et al. 1990; Chakraborty et al. 2010]. ECC/replication hybrids have also been composed in both the cache domain [Zhang et al. 2003] and the SPM domain [Li et al. 2005]. Since we target SPM based systems, the closest piece of work to E-RoC is the work done in [Li et al. 2005], which uses parity calculations to check the validity of data; in the case of an error, an extra copy of the data is fetched, assuming no errors in the extra. Data mapped onto the E-RAID can follow the same process as proposed in [Panda et al. 1997; Verma et al. 2003; Issenin et al. 2004; Issenin et al. 2006; Bathen et al. 2008; Cho et al. 2008; Bathen et al. 2009] with a few minor modifications, mainly E-RAID configuration requests before data is mapped onto the E-RAID. Because E-RoC relies on data duplication and simple comparisons/XORs to check/correct data, we have seen great improvements not only in power but also in performance, as most ECC/replication approaches require expensive ECC parity calculations on every transaction. Similarly, E-RoC's space overhead is very similar to existing data replication techniques as it keeps at most two copies of the data and the parity (E-RAID 1+P and TMR).

Most SPM approaches [Panda et al. 1997; Kandemir et al. 2001; Verma et al. 2003; Suhendra et al. 2006; Suhendra et al. 2008; Gauthier et al. 2010; Takase et al. 2010; Pyka et al. 2007] focus on single SPM management through static analysis and profiling information. [Shalan and Mooney 2000] looked at dynamic memory management for global memory through the use a hardware module. [Francesco et al. 2004] proposed a memory manager that supports dynamic allocation of SPM space, which supports block-based allocation (fixed and variable). Egger et al. proposed SPM management techniques for MMU supported [Egger et al. 2008] and MMU-less embedded

systems [Egger et al. 2010], where code was divided into cacheable code and pageable (SPM) code, and the most commonly used code is mapped onto SPM space. [Pyka et al. 2007] introduced an OS-level management layer that exploited hints from static analysis at run-time to dynamically map objects onto SPMs.

Our approach is the first piece of work that looks at the idea of dynamic allocation in *distributed* SPMs, as well as their use from a reliability perspective. Our approach can be complemented by existing SPM management approaches [Panda et al. 1997; Kandemir et al. 2001; Verma et al. 2003; Suhendra et al. 2006; Suhendra et al. 2008; Bathen et al. 2009; Gauthier et al. 2010; Takase et al. 2010; Pyka et al. 2007] and even exploit some of the allocation policies presented in [Francesco et al. 2004].

Our approach is capable of exploiting the built-in circuitry in on-chip memories, and enhance their fault tolerance by building the E-RAID levels on top. Moreover, our approach is dynamic enough so that you can have a fully heterogeneous E-RAID environment, each meeting different fault tolerance, power, and performance needs for each application.

Finally, this work is different from all existing SPM management and memory reliability approaches in that it is the *first* to propose a system-level solution for both efficient on-chip distributed SPM management and exploiting said distribution to provide a fully distributed fault tolerant scheme.

6. EXPERIMENTAL EVALUATION

6.1. Experimental Goals

The goals of the experimentation section are to highlight the many benefits of having an E-RAID enabled memory subsystem. First, we compare our approach with state-of-the-art approaches. Second, we show the effects of the load of the system on E-RoC's performance. Third, we evaluate the effects of different bus-arbitration policies on E-RoC's performance. Fourth, we look at how E-RoC and the E-RAID concepts can be applied and used to exploit (enhance) traditional ECC schemes and evaluate their efficiency in terms of Yield, Dynamic/Static Energy, and Performance. Fifth, we look at the effects of having limited E-RAID space for a given application on performance/power consumption. Finally, we explore the notion of creating heterogeneous E-RAID systems and show how an application can benefit from partitioning its data and mapping it to the right E-RAID level.

6.2. Experimental Setup

We implemented our E-RAID ideas in the Chip-Multiprocessor E-RoC Manager. The E-RoC concept has been implemented in SystemC [OSCI 2005] and embedded into our SystemC based modeling framework [Pasricha 2002; Bathen et al. 2009; Bathen et al. 2009], which allows us to estimate both power and performance for the entire system. Our framework allows us to interface with CACTI [Thoziyoor et al. 2004], voltage scale our memories and observe the effects of the scaling (e.g., increased access latency, uniform error distribution, etc.). Since we deal with SPM based systems, we compare our work with two existing approaches: (i) a standard ECC based approach (SECDED) that verifies and corrects the data (labeled ECC), and (ii) the duplication with parity work [Li et al. 2005] (labeled DUP), we explored various CMP configurations and mapped a series of benchmarks from [Hara et al. 2008; Lee et al. 1997]. We implemented various ECC schemes presented in [Chen and Hsiao 1984; Kim et al. 2007]. Our approach can be used for protection against both soft and hard errors, however, for this work we will focus solely on process variation induced errors, and we base our fault models on the work presented in [Makhzan et al. 2007; Jahinuzzaman et al. 2008; Kalter et al. 1990] for the various voltages used.

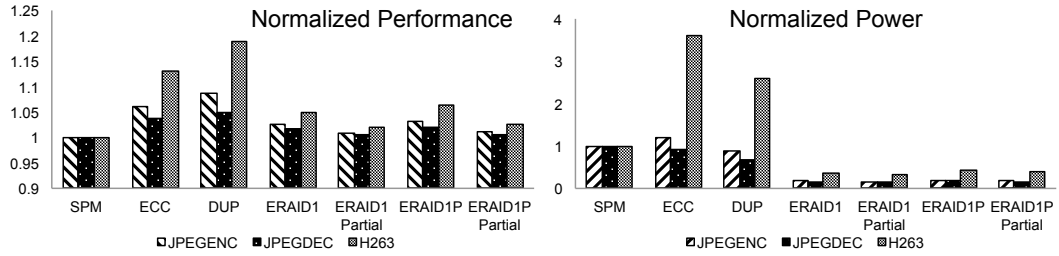

Fig. 16. Normalized performance (left) and power consumption (right) for three different benchmarks.

6.3. Normalized Performance and Power Consumption

Figure 16 shows the normalized performance and power consumption for a CMP with 8 cores and 8 4KB SPMs. For this experiment we configured the E-RoC manager as shown in Figure 9 (a). The base case is a CMP with no voltage scaling applied to the SPMs. The standard SPM case outperforms all other configurations because a) the ECC/parity check overheads incurred in the ECC/DUP cases and b) the backend address translation as well as fetching the data from up to three different DSPAMs (in case data needs to be reconstructed) performed by E-RAID. Our E-RAID systems outperforms the ECC/DUP configurations by up to 14% and consumes up to 80% less power than the standard high voltage SPM CMP and up to 85% less power than the ECC/DUP approaches.

6.4. Selective Data Partitioning

E-RoC is well suited for approaches that can selectively partition data into critical/vulnerable and non-critical data [Lee et al. 2006]. Such approaches allow us to further reduce power consumption and improve performance as E-RoC offers the ability to choose low power policies such as NO E-RAID, where non-critical (e.g., image pixel) data may be mapped. This is illustrated in Figure 16 (labeled: Partial in the graphs), where by selectively creating E-RAID systems (i.e., mapping critical data to E-RAID1 space and non-critical to NO E-RAID space), performance can be improved by up to 5%, and power consumption can be further reduced by up to 15%.

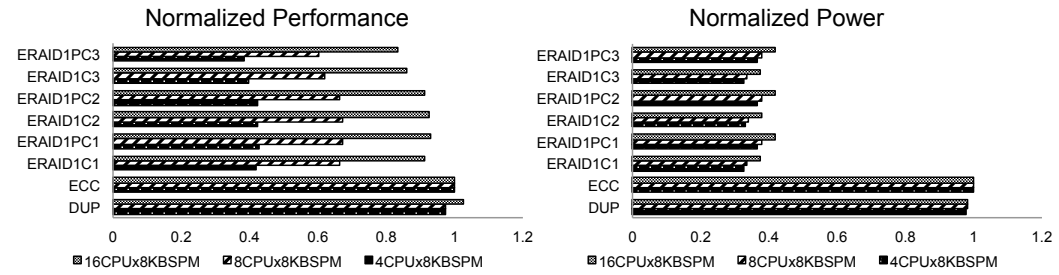

Fig. 17. Normalized performance (left) and power consumption (right) for a pipelined JPEG Encoder using different platform configurations with E-RoC manager support as shown in Figure 9 .

6.5. Choosing the Right Platform Configuration

The platform configuration affects the systems performance and power consumption footprint. Figure 17 shows the normalized performance and power consumption for different CMP configurations, where C1-3 refer to configurations (a-c) from Figure 9. As expected, performance was greatly improved when migrating from configuration C1 (shared bus) to C2 (dual bus) due to less bus contention. Configuration C3 (stand alone E-RoC), was able to further improve performance by 10% with respect to C1. Power consumption remains within a 2% difference for all three configurations (C1-3). This pipelined implementation of JPEG showed up to 61% latency reduction and 67% power reduction when compared to standard ECC/DUP approaches.

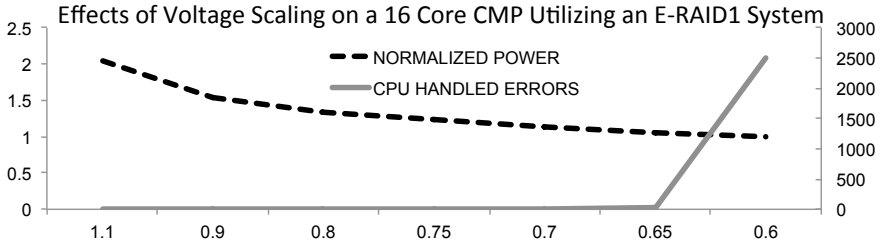

Fig. 18. Effects of Voltage Scaling on E-RoC.

6.6. Effects of Voltage Scaling on E-RoC

Figure 18 shows the effects of voltage scaling on a 16 Core CMP with a total of 16 concurrently managed E-RAID systems. As we scale down voltage, power consumption is indeed being reduced (dashed line), while the error rate skyrockets (straight line). This behavior confirms our initial observations from Section 4.1 that aggressive voltage scaling of SPMs increases the memory error rate (handled by our E-RAID systems) but reduces the power consumption.

6.7. Pipelined JPEG Encoder Performance and Power Consumption

For this experiment we took the JPEG Encoder and pipelined its execution [Bathen et al. 2009] in order to fully utilize the system's resources. The goal is to evaluate the effects of the workload on the memory subsystem.

Figure 19 (a) shows the performance improvements for multiple configurations (C1-3 refer to the configurations proposed in Section 4.4), as well as different E-RAID levels over the ECC/DUB techniques. As we can observe, performance improvements are higher when we move away from the shared bus model as well as when the number of masters concurrently accessing the bus, memory subsystem and E-RoC is reduced. This can be verified by looking at the behavior of the CMP with 16 Cores and the CMP with 4 cores.

Figure 19 (b) shows the power reduction achieved through E-RoC. On average, over system wide 55% power consumption has been reduced as our E-RAID levels have been optimized to minimize both power consumption and performance degradation due to our redundant memory subsystem. As it was the case with performance, as the number of masters accessing E-RoC increases, both power and performance improvements remain moderate.

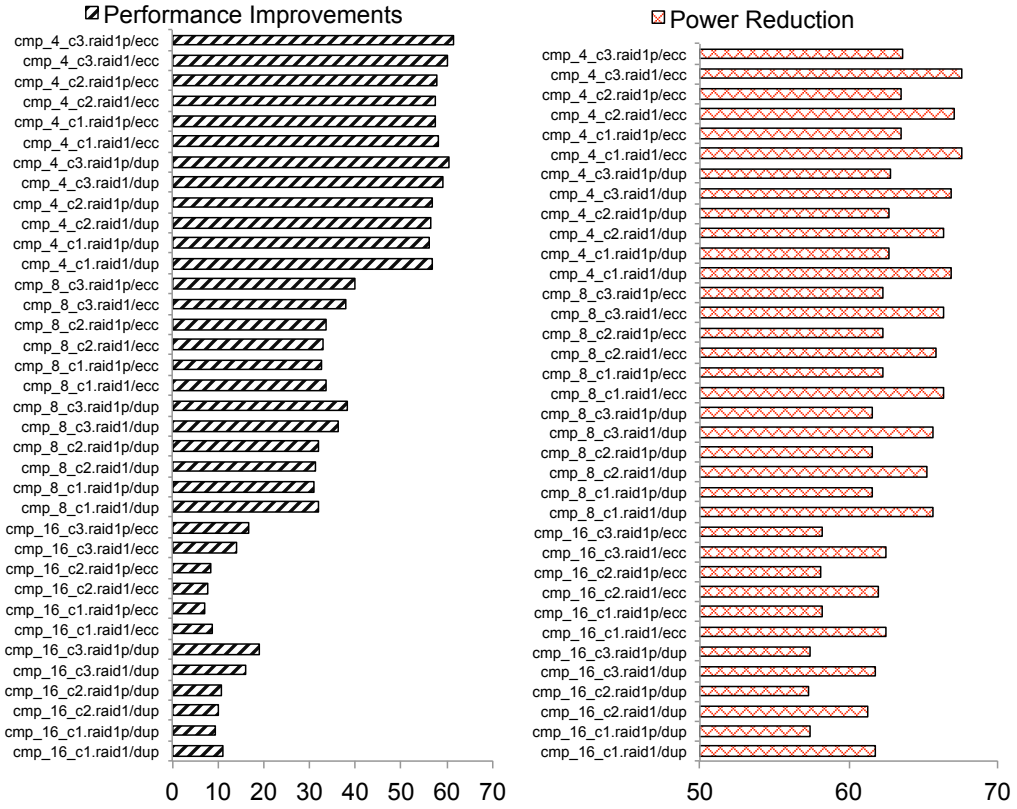

Fig. 19. Improvements over Traditional Approaches.

6.8. Effects of Arbitration Policy

Arbitration policies will have a major impact in E-RoCs performance and power consumption, as shown in Figure 20 (a), (b), respectively. The reason behind this dependence is the fact that E-RoC serves both as a slave to all the CPUs/masters in the system who need a reliable memory system and a master to all of its managed DSPAMS. As we can see from Figure 20, E-RoC suffers degradation in both power and performance when a round-robin arbitration scheme is used in configuration 1 (C1 in Figure 9 (a)) as E-RoC must compete with all other masters for bus cycles. This overhead is minimized when we move away from the single shared bus model, as shown in configurations 2 and 3, Figure 9 (b), and (c) respectively.

6.9. Comparison Among Various Schemes

Figures 21, 22, 23, and 24 show the Yield, Dynamic Power Consumption, Static Power Consumption, and Performance Overheads for the various fault tolerant schemes shown in Table I. For this set of experiments we varied the voltage for each of the dedicated DSPAMs allocated to our E-RoC manager. We set the configuration to the bus-based CMP (Figure 9 (a)), with one active processing core and a total of eight 8KB SPMs. Each SPM was aggressively voltage scaled, ranging from 0.35 Vdd to 1.1 (Nominal) Vdd per [Jahinuzzaman et al. 2008]. We then mapped a set of benchmarks from [Hara et al. 2008; Lee et al. 1997] assuming no data cache, and all data either being

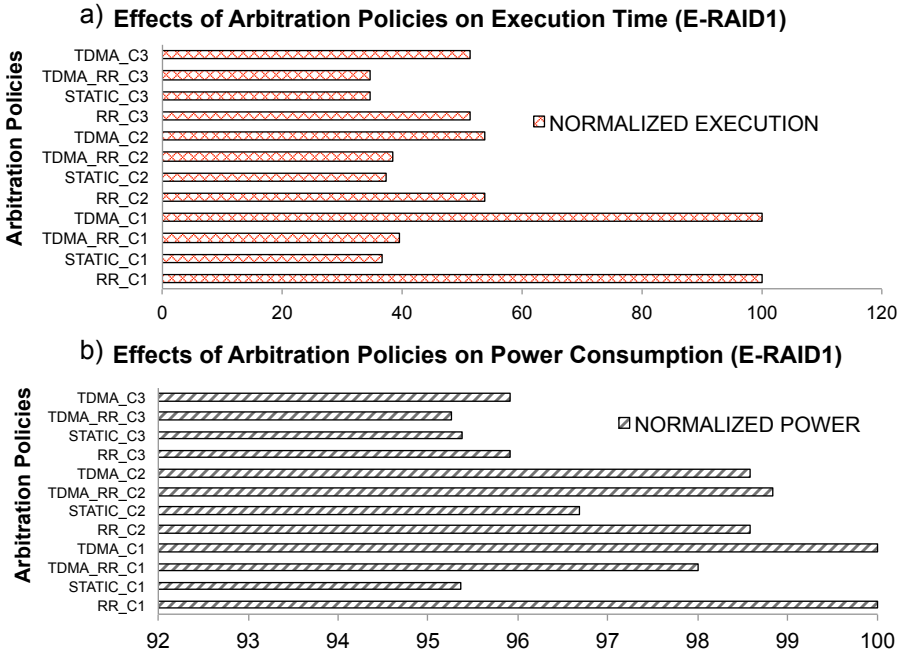

Fig. 20. Effects of Arbitration Policy.

mapped onto SPMs (Logical SPMs with E-RAID support), or main memory. All results are normalized to the case where the SPMs are running at Nominal Vdd (1.1 Vdd), and do not have any error protection mechanism in place. We denote the baseline as *Nominal*. Note that for all approaches, the same amount of physical space is available, the only difference is the usable space which is determined by the number of Logical SPMs created and their respective E-RAID levels (if any). The goal here is to show how our framework can be used by designers to investigate through tradeoff analysis (yield, power, performance) which E-RAID levels would make sense for their given applications (sets of applications).

For this experiment we assumed that in the case of error detection schemes (EDC, EDC8, E-RAID1, etc.), we could fetch the correct copy from main memory at the cost of the extra power/performance overhead. Because of this, most schemes achieve near 100% Yield around 0.5 - 0.65 Vdd. On average, we see that our E-RAID levels converge to 100% Yield much faster than traditional ECC approaches. Moreover, E-RAID levels that exploit ECC (e.g., E-RAID ECC + 1, E-RAID RP + ECC) can guarantee 99.9% Yield at ultra low Vdd on average, where as SECDED and DECTED were able to attain 99.1% and 99.4% Yield respectively. Our E-RAID levels (detection and correction) achieved a worst case 93.9% Yield (AES), where as the traditional ECC approaches achieved a worst case of 34.1% Yield (MOTION).

Dynamic power consumption can be observed in Figure 22. As expected, at ultra low Vdd, the error rate is so high that on error detection, many schemes fetch data from off-chip memory, so the increased off-chip traffic offsets the energy savings due to the voltage scaling and this can be observed by the first set of points in the graphs where the power consumption with respect to the Nominal case is much higher. As we start increasing the voltage we start seeing a decline in dynamic power consumption

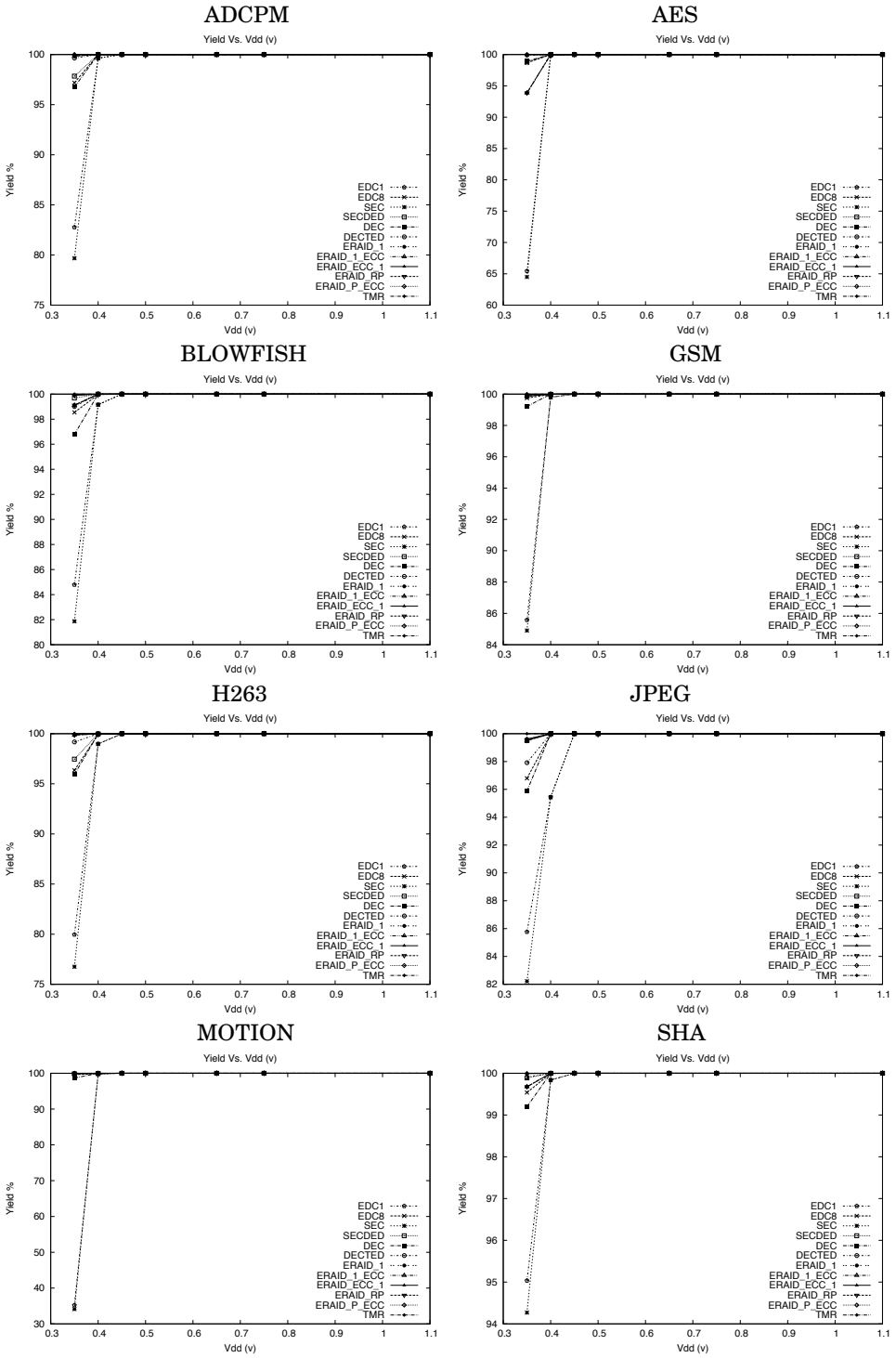

Fig. 21. Yield Comparison

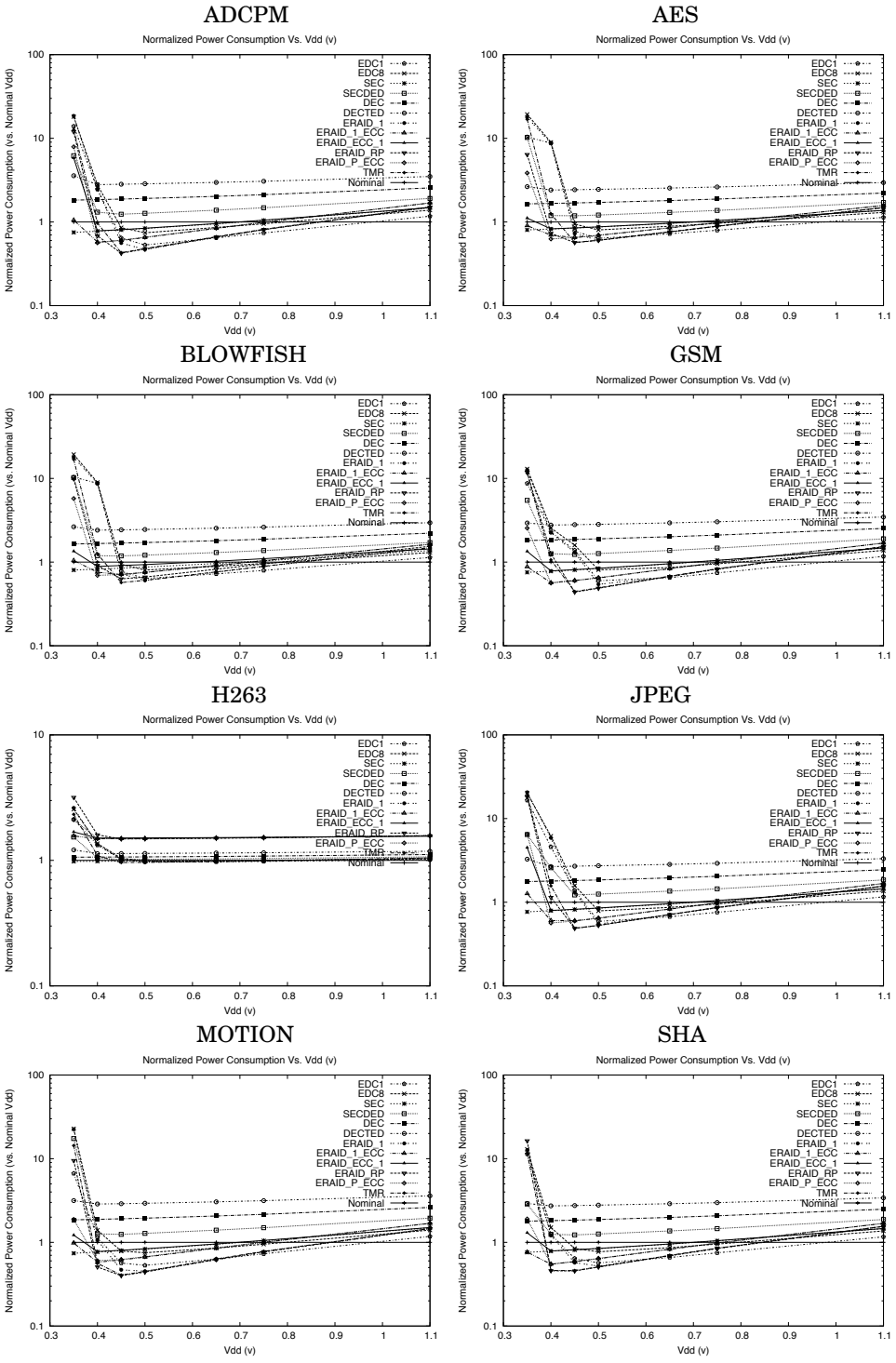

Fig. 22. Normalized Dynamic Power Consumption Comparison

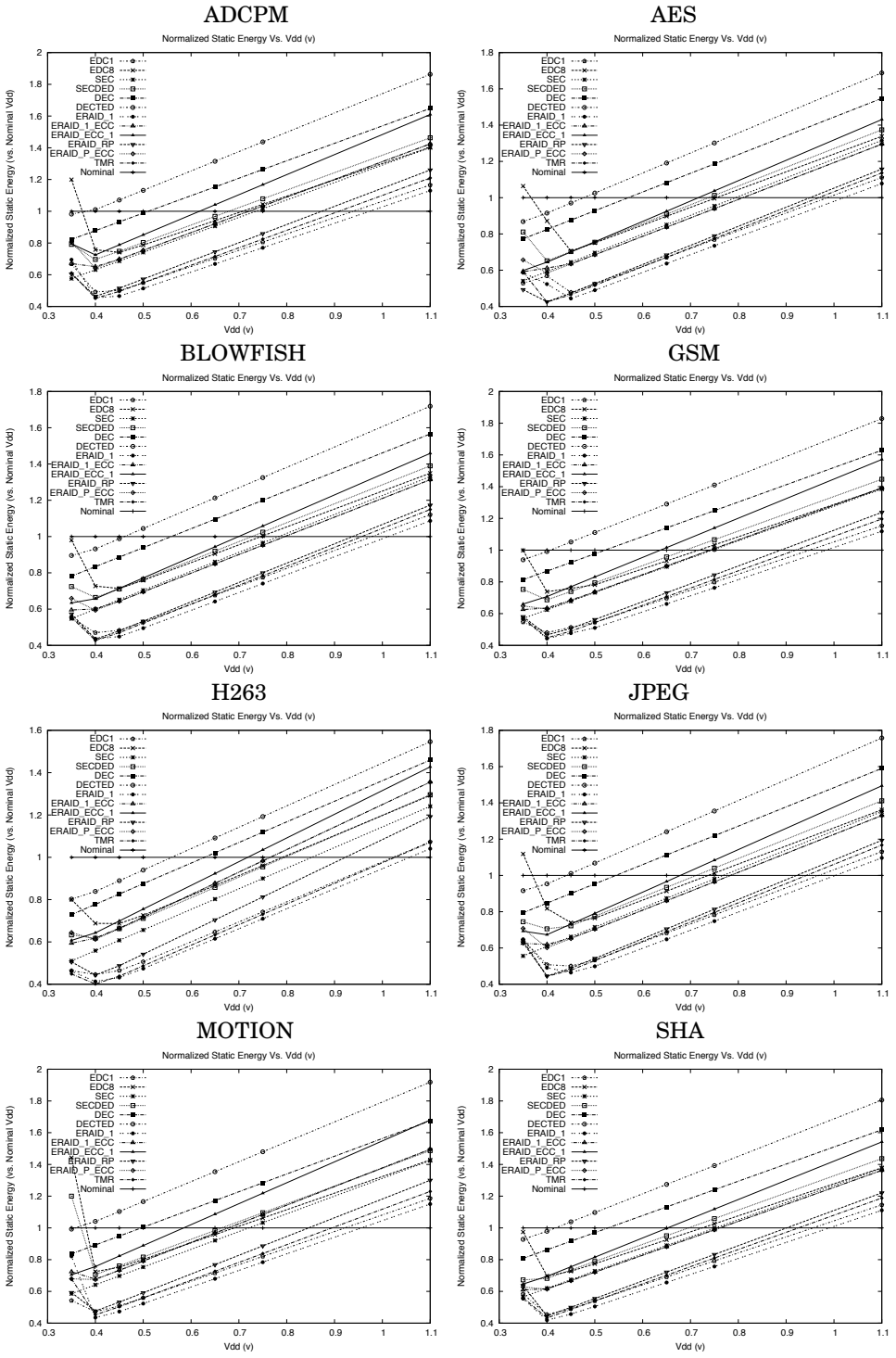

Fig. 23. Normalized Static Energy Comparison

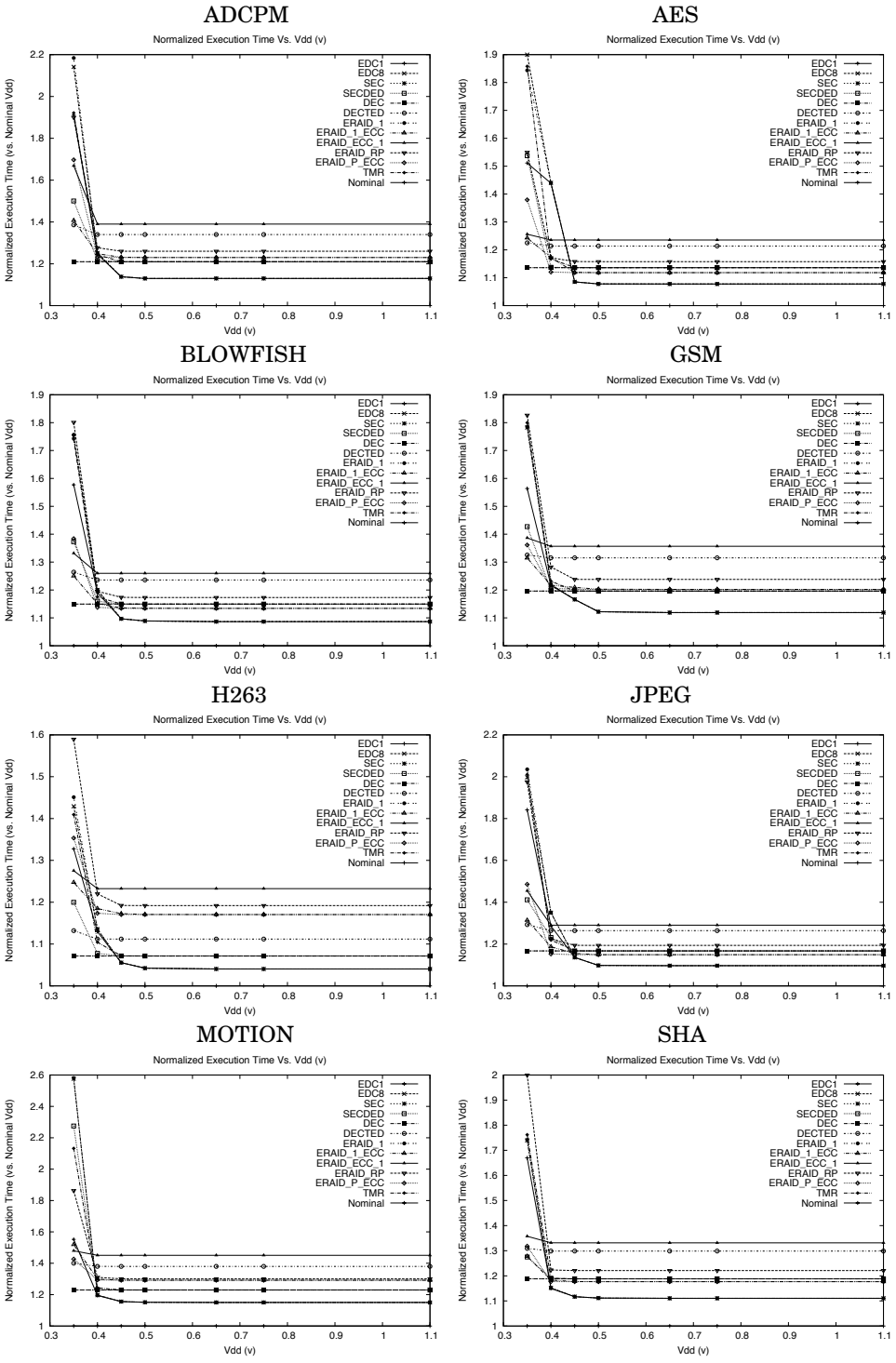

Fig. 24. Normalized Execution Time Comparison

and find that for these set of experiments, the sweet spot for our E-RAID schemes is between 0.4 and 0.5 Vdd. As we start increasing the voltage further (0.8+ Vdd) we see that the extra accesses to the DSPAMs (SPMs) by our E-RAIDs start to impact the power consumption of the system, and thus, the power consumption is on the incline. These set of graphs show how our framework would allow designers to explore various workloads and customize the E-RAID levels that would meet their power budget. H263 is the one example where it is difficult to observe a sweet spot in terms of power consumption, this is due to the fact that H.263 incurred many off-chip memory accesses, which led the off-chip traffic to dominate system power consumption, however, a much smarter allocation scheme that exploits data reuse may be used to reduce the off-chip traffic [Panda et al. 1997; Verma et al. 2003; Issenin et al. 2004; Issenin et al. 2006; Bathen et al. 2008; Cho et al. 2008; Bathen et al. 2009]. DECTED achieves the highest Yield from all the ECC approaches at the cost in both power consumption and performance, as we can see, no single instance shows power consumption savings for DECTED even with the voltage scaling of SPM's data banks. We observe an average of 22% dynamic power consumption increase by using traditional ECC approaches (EDC1, EDC8, SEC, SECDED, DEC, DECTED), where as we observe average savings of 27% for our E-RAID schemes (E-RAID 1, E-RAID 1 + ECC, E-RAID ECC + 1, E-RAID RP, E-RAID RP + ECC, E-RAID TMR).

Static energy for various applications can be observed in Figure 23. In our experimental setup, all DSPAMs to our E-RoC manager (physical SPM space to be used for Logical SPMs and E-RAIDs) are voltage scaled, if the memories have a built-in ECC chip, then the banks holding the ECC parities are not voltage scaled, but their data banks are. So for each of the different schemes in Figure 23 we observe different static power consumption. As with dynamic power consumption, static energy is high at ultra low Vdd (0.35) because of the many off-chip accesses that increases the execution time of the application. Just line with dynamic power, we can find the sweet spot to be around 0.4 to 0.45 Vdd for these set of applications. Again, this set of examples show how our approach can be used to find the right E-RAID level to save both static and dynamic power. We observe that E-RAID levels with no ECC (SEC) support (E-RAID 1, E-RAID TMR) consume the less amount of static power, mainly because all of their data banks are voltage scaled, where as any scheme with ECC support must not voltage scale the banks that contain the parities. We see that on average traditional ECC approaches are able to save static energy by 6.4%, where as our E-RAID approaches achieve 23.4% static energy savings.

The dynamic and static power consumption savings come at a cost however. Figure 24 shows the performance overheads for the various fault tolerant schemes. We observe that for ultra-low Vdd most schemes pay the price of going off-chip, and like with power consumption (energy), the off-chip memory accesses greatly influence the performance of the system (increased execution time, increased power consumption, etc.). As we slightly increase the voltage, we observe that our E-RAID schemes are on-par with simple error detection approaches (with respect to increased latency), and greatly outperform the error-correction approaches (SECDED, DECTED). We observe latency increase from 4% (EDC1 0.65 Vdd+ / H.263) to 61% (E-RAID 1 0.35 Vdd / MOTION). We observe that on average our approaches (E-RAID 1, E-RAID 1 + ECC, E-RAID ECC + 1, E-RAID RP, E-RAID RP + ECC, E-RAID TMR) incur 2% higher overheads than traditional ECC approaches (EDC1, EDC8, SEC, SECDED, DEC, DECTED). We observe that for Vdd above 0.45, on average, our E-RAID levels with error correction support (SEC) incur 3% lower overheads over the more traditional SECDED/DECTED schemes. On average, for higher Vdd, EDC1, EDC8, and E-RAID 1 achieve the lowest overheads, this is because they only need a simple XOR or AND to check the correct-

ness of the data. Note that, at low Vdd, EDC1 and EDC8 achieve the poorest Yield of all the schemes we tested.

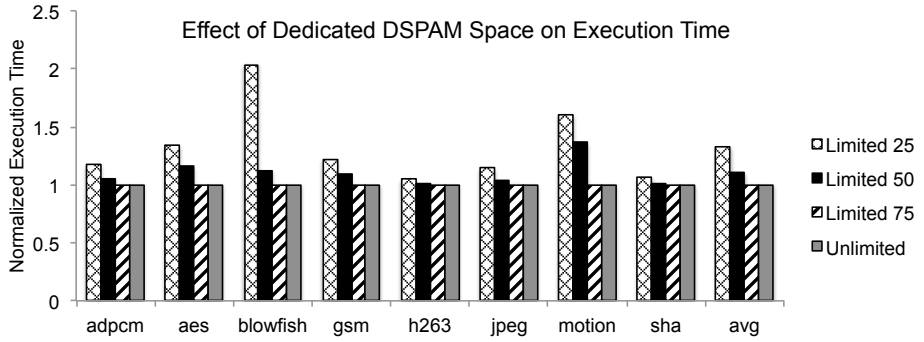

Fig. 25. Effects of Limited DSPAM Space on Performance.

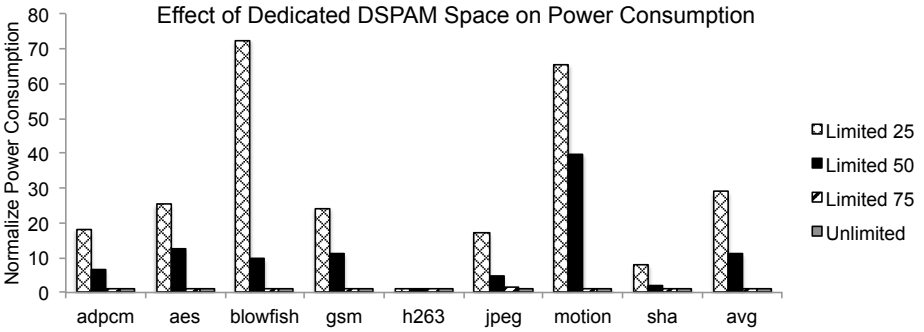

Fig. 26. Effects of Limited DSPAM Space on Power Consumption.

6.10. Effects of Limited DSPAM Space

For this experiment we wanted to evaluate the the effect of having limited on-chip space for a series of applications. The goal was to investigate what would happen if our allocation algorithms were unable to accommodate the application's data in Logical SPMs (e.g., the DSPAM space became too fragmented or there are already too many Logical SPMs with their respective E-RAID levels created/allocated). Our basic configuration consists of a single active core, the E-RoC manager, and a series of 8KB DSPAMs. We varied the amount of data we can map to our Logical SPMs (25% to full allocation), and normalized the execution time/power consumption to the case where we have unlimited resources (full allocation). For this experiment, we created a Logical SPM to hold a given percentage (25%, 50%, 75%, 100%) of SPM mappable data with an E-RAID 1 level. From Figure 25 and Figure 26 we can see that on average, latency increases by 24% and dynamic power consumption increases by 96% due to the extra off-chip memory accesses. Similarly, we observe 9% latency increase and 90% dynamic power consumption increase on average with 50% of data mapped to our Logical SPM.

Finally, we see that if we further increase the amount of data mapped to our Logical SPMs from 50% to 75% we greatly reduce both, power consumption and latency overheads to 9% and 0.1% respectively.

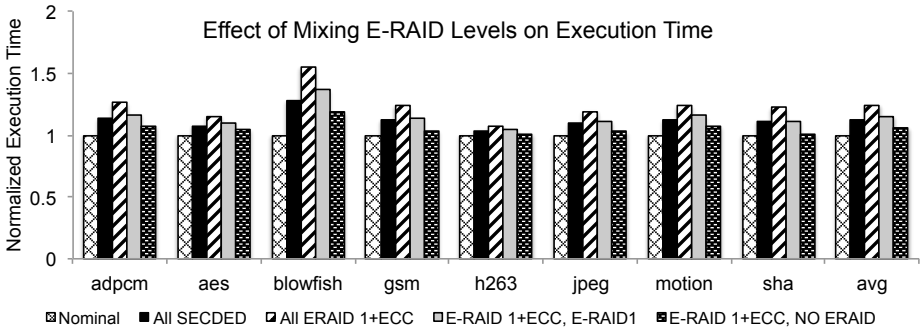

Fig. 27. Effects of Mixing E-RAID's on Performance.

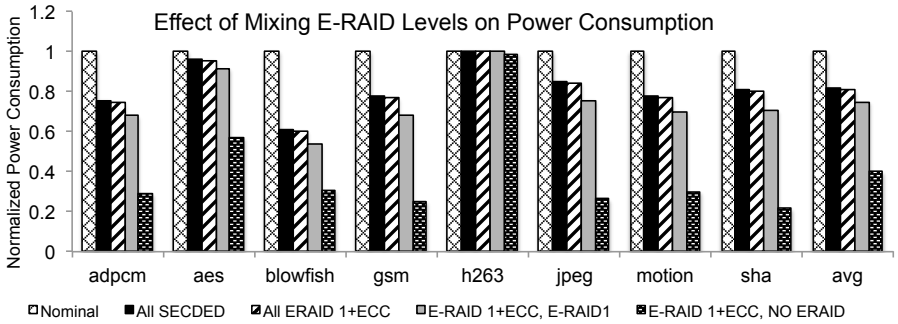

Fig. 28. Effects of Mixing E-RAID's on Power Consumption.

6.11. Exploiting E-RAID's Flexibility

For this experiment we wanted to evaluate the benefits of mixing E-RAID levels for a series of applications. The baseline in this experiment consists of an 8KB SPM running at nominal Vdd with a single core and the E-RoC manager. We have four voltage scaled (Vdd = 0.5 V, which is around the sweet spot discussed in Section 6.9) configurations: 1) ALL SECDED, which means that the on-chip memory space is protected by SECDED, 2) ALL E-RAID 1 + ECC, which means that the entire on-chip memory space is protected by an E-RAID 1 + ECC level, 3) E-RAID 1 + ECC, E-RAID 1, which means that we have created two separate Logical SPMs, each Logical SPM is associated with a different E-RAID Level (one has E-RAID 1 + ECC, the other one has E-RAID 1), 4) E-RAID 1 + ECC, NO E-RAID, which means that we have two Logical SPMs, one protected by an E-RAID 1 + ECC level and the other by a NO E-RAID level. Note that though the logical partitioning of the memory space is different, the physical space is the same for all cases and the same data is mapped to on-chip and off-chip memory

space. The only difference is the level/scheme protecting the data. This type of partitioning follows the notion of critical data (e.g., control variables) and non-critical data (e.g., data pixels) discussed in [Lee et al. 2006]. As we can see from Figures 27 and 28, the ALL SECDED scheme incurs an average 10.7% increase in execution time while saving an average of 22% dynamic power with respect to the Nominal case. The ALL E-RAID 1 + ECC scheme incurs a higher increase in execution time (19% on average) and higher power consumption savings (23%). The third scheme (E-RAID 1 + ECC, E-RAID 1) incurs a lower overhead (12%) than the second scheme (ALL E-RAID 1 + ECC) while further reducing power consumption (an average reduction of 33% over the Nominal case). Finally, the fourth scheme (E-RAID 1 + ECC, NO E-RAID), which shows the true power of mixing E-RAID levels is able to reduce the dynamic power consumption by 150% (note that NO E-RAID is pure low voltage storage, with no extra protection) at the cost of an average 5% increase in execution time. This all translates to 9.8% energy savings for the first approach, 0.72% energy savings for the second approach, 17% energy savings for the third approach, and 138% energy savings for the fourth approach.

7. CONCLUSIONS AND FUTURE WORK

In this paper we introduced the notions of Embedded RAID (E-RAID) and Embedded RAIDs-on-Chip (E-RoC), a distributed dynamically managed reliable memory subsystem. Among the key concepts introduced are: the notion of reliability via redundancy using an E-RAID system; a set of E-RAID levels that are optimized for use in embedded SoCs; and the concept of distributed dynamic scratch pad allocatable memories (DSPAMs) and their allocation policies. We exploited aggressive voltage scaling to reduce power consumption overheads due to parallel DSPAM accesses. We defined the first proof-of-concept E-RoC manager that exploits these ideas. We presented a set of architectural designs by which E-RoC based systems can be configured. We explored the flexibility and benefits of Embedded RAIDs-on-Chip by 1) studying their ability to complement existing fault tolerant approaches (e.g., ECC), 2) their ability to create a heterogeneous E-RAID level environment to match the different fault tolerance needs of each application, 3) the effects of arbitration policies, and 4) the power consumption/energy and performance overheads of various E-RAID levels.

Our experimental results on multimedia benchmarks show that E-RoC's fully distributed redundant reliable memory subsystem can attain up to 85% in power consumption reduction, and up to 61% latency reduction due to error checks/corrections. On average, we see that our E-RAID levels converge to 100% Yield much faster than traditional ECC approaches. Moreover, E-RAID levels that exploit ECC (e.g., E-RAID ECC + 1, E-RAID RP + ECC) can guarantee 99.9% Yield at ultra low Vdd on average, where as SECDED and DECTED were able to attain 99.1% and 99.4% Yield respectively. Our E-RAID levels (detection and correction) achieved a worst case 93.9% Yield (AES), where as the traditional ECC approaches achieved a worst case of 34.1% Yield (MOTION). We observe an average of 22% dynamic power consumption increase by using traditional ECC approaches (EDC1, EDC8, SEC, SECDED, DEC, DECTED), where as we observe average savings of 27% for our E-RAID schemes (E-RAID 1, E-RAID 1 + ECC, E-RAID ECC + 1, E-RAID RP, E-RAID RP + ECC, E-RAID TMR). We see that on average traditional ECC approaches are able to save static energy by 6.4%, where as our E-RAID approaches achieve 23.4% static energy savings. We observe that on average our approaches (E-RAID 1, E-RAID 1 + ECC, E-RAID ECC + 1, E-RAID RP, E-RAID RP + ECC, E-RAID TMR) incur 2% higher overheads than traditional ECC approaches (EDC1, EDC8, SEC, SECDED, DEC, DECTED). We observe that for Vdd above 0.45, on average, our E-RAID levels with error correction support (SEC) incur 3% lower overheads over the more traditional SECDED/DECTED schemes. Finally, we

observe that mixing E-RAID levels allows us to reduce the dynamic power consumption by 150% at the cost of an average 5% increase in execution time over traditional approaches.

Since E-RoC is a first of its kind piece of work, there are many opportunities for further optimization that we are currently exploring: E-RAID allocation policies might lead to DSPAM space fragmentation. Similarly, the shared bus model drives E-RoC performance down, thereby motivating the use of more complex communication fabrics (i.e., Bus Matrix and NoCs).

REFERENCES

ANGIOLINI, F., ATIENZA, D., MURALI, S., BENINI, L., AND DE MICHELI, G. 2006. Reliability support for on-chip memories using networks-on-chip. In *Int. Conf. on Computer Design (ICCD) 2006.*

BAI, K. AND SHRIVASTAVA, A. 2010. Heap data management for limited local memory (llm) multi-core processors. In *Proceedings of the eighth IEEE/ACM/IFIP Int. Conf. on Hardware/software codesign and system synthesis*. CODES/ISSS '10. 317–326.

BATHEN, L., AHN, Y., DUTT, N., AND PASRICHA, S. 2009. Inter-kernel data reuse and pipelining on chip-multiprocessors for multimedia applications. In *Embedded Systems for Real-Time Multimedia, 2009. ESTIMedia 2009. IEEE/ACM/IFIP 7th Workshop on*.

BATHEN, L. AND DUTT, N. 2011a. E-roc: Embedded raids-on-chip for low power distributed dynamically managed reliable memories. In *Design, Automation Test in Europe Conf. Exhibition (DATE), 2011.*

BATHEN, L. A. D., AHN, Y., PASRICHA, S., AND DUTT, N. D. 2009. A methodology for power-aware pipelining via high-level performance model evaluations. In *Proceedings of the 2009 10th International Workshop on Microprocessor Test and Verification*. MTV '09. 19–24.

BATHEN, L. A. D. AND DUTT, N. 2011b. Tustgem: Dynamic trusted environment generation for chip-multiprocessors. *to appear, Proceedings of the 2011 IEEE Int. Symposium on Hardware-Oriented Security and Trust*.

BATHEN, L. A. D., DUTT, N. D., AND PASRICHA, S. 2008. A framework for memory-aware multimedia application mapping on chip-multiprocessors. In *ESTImedia*. 89–94.

CHAKRABORTY, A., HOMAYOUN, H., KHAJEH, A., DUTT, N., ELTAWIL, A., AND KURDAHI, F. 2010. $e = mc^2$: less energy through multi-copy cache. In *Proceedings of the 2010 international conference on Compilers, architectures and synthesis for embedded systems*. CASES '10. 237–246.

CHEN, C. L. AND HSIAO, M. Y. 1984. Error-correcting codes for semiconductor memory applications: a state-of-the-art review. *IBM J. Res. Dev. 28*, 124–134.

CHO, D., PASRICHA, S., ISSENIN, I., DUTT, N., PAEK, Y., AND KO, S. 2008. Compiler driven data layout optimization for regular/irregular array access patterns. In *Proceedings of the 2008 ACM SIGPLAN-SIGBED Conf. on Languages, compilers, and tools for embedded systems*. LCTES '08. 41–50.

DJAHROMI, A. K., ELTAWIL, A. M., KURDAHI, F. J., AND KANJ, R. 2007. Cross layer error exploitation for aggressive voltage scaling. In *Proceedings of the 8th Int. Sym. on Quality Electronic Design*. ISQED '07. IEEE Computer Society, Washington, DC, USA, 192–197.

EGGER, B., KIM, S., JANG, C., LEE, J., MIN, S. L., AND SHIN, H. 2010. Scratchpad memory management techniques for code in embedded systems without an mmu. *Computers, IEEE Trans. on 59,* 8.

EGGER, B., LEE, J., AND SHIN, H. 2008. Dynamic scratchpad memory management for code in portable systems with an mmu. *ACM Trans. Embed. Comput. Syst.* 7.

FRANCESCO, P., MARCHAL, P., ATIENZA, D., BENINI, L., CATTHOOR, F., AND MENDIAS, J. M. 2004. An integrated hardware/software approach for run-time scratchpad management. In *Proceedings of the 41st annual Design Automation Conf.* DAC '04.

GAUTHIER, L., ISHIHARA, T., TAKASE, H., TOMIYAMA, H., AND TAKADA, H. 2010. Minimizing inter-task interferences in scratch-pad memory usage for reducing the energy consumption of multi-task systems. In *Proceedings of the 2010 Int. Conf. on Compilers, architectures and synthesis for embedded systems*. CASES '10. 157–166.

GHOSH, S., BASU, S., AND TOUBA, N. 2004. Reducing power consumption in memory ecc checkers. In *Test Conf., 2004. Proceedings. ITC 2004. Int.*

HARA, Y., TOMIYAMA, H., HONDA, S., TAKADA, H., AND ISHII, K. 2008. Chstone: A benchmark program suite for practical c-based high-level synthesis. In *Circuits and Systems, 2008. ISCAS 2008. IEEE Int. Sym. on*. 1192 –1195.

IBM. 2005. The cell project. *IBM, http://www.research.ibm.com/cell/*.

INTEL. 2007. Teraflops research chip. *Intel, http://techresearch.intel.com/ProjectDetails.aspx?Id=151*.

INTEL. 2009. Single-chip cloud computer. *Intel, http://techresearch.intel.com/ ProjectDetails.aspx?Id=1*.

ISSENIN, I., BROCKMEYER, E., DURINCK, B., AND DUTT, N. 2006. Multiprocessor system-on-chip data reuse analysis for exploring customized memory hierarchies. In *Proceedings of the 43rd annual Design Automation Conf.* DAC '06. 49–52.

ISSENIN, I., BROCKMEYER, E., MIRANDA, M., AND DUTT, N. 2004. Data reuse analysis technique for software-controlled memory hierarchies. In *DATE*. 202–207.

JAHINUZZAMAN, S., SHAKIR, T., LUBANA, S., SHAH, J., AND SACHDEV, M. 2008. A multiword based high speed ecc scheme for low-voltage embedded srams. In *Solid-State Circuits Conf., 2008. ESSCIRC 2008. 34th European*.

JUNG, S. C., SHRIVASTAVA, A., AND BAI, K. 2010. Dynamic code mapping for limited local memory systems. In *Application-specific Systems Architectures and Processors (ASAP), 2010 21st IEEE Int. Conf. on*. 13 –20.

KALTER, H. L., STAPPER, C. H., BARTH, J. E., DILORENZO, J., DRAKE, C. E., FIFIELD, J. A., KELLEY, G. A., LEWIS, S. C., VAN DER HOEVEN, W. B., AND YANKOSKY, J. A. 1990. A 50-ns 16-mb dram with a 10-ns data rate and on-chip ecc. *IEEE Journal of Solid-state Circuits 25*, 1118–1128.

KANDEMIR, M., RAMANUJAM, J., IRWIN, J., VIJAYKRISHNAN, N., KADAYIF, I., AND PARIKH, A. 2001. Dynamic management of scratch-pad memory space. In *Proceedings of the 38th annual Design Automation Conf.* DAC '01.

KIM, J., HARDAVELLAS, N., MAI, K., FALSAFI, B., AND HOE, J. C. 2007. Multi-bit error tolerant caches using two-dimensional error coding. In *MICRO*. 197–209.

KIM, S. 2006. Area-efficient error protection for caches. In *Proceedings of the conference on Design, automation and test in Europe: Proceedings*. DATE '06. European Design and Automation Association, 3001 Leuven, Belgium, Belgium, 1282–1287.

KURDAHI, F., ELTAWIL, A., YI, K., CHENG, S., AND KHAJEH, A. 2010. Low-power multimedia system design by aggressive voltage scaling. *Very Large Scale Integration (VLSI) Systems, IEEE Trans. on 18,* 5.

LEE, C., POTKONJAK, M., AND MANGIONE-SMITH, W. H. 1997. Mediabench: a tool for evaluating and synthesizing multimedia and communicatons systems. In *Proceedings of the 30th annual ACM/IEEE Int. Sym. on Microarchitecture*. MICRO 30. IEEE Computer Society, Washington, DC, USA, 330–335.

LEE, K., SHRIVASTAVA, A., ISSENIN, I., DUTT, N., AND VENKATASUBRAMANIAN, N. 2006. Mitigating soft error failures for multimedia applications by selective data protection. In *Proceedings of the 2006 Int. Conf. on Compilers, architecture and synthesis for embedded systems*. CASES '06.

LI, F., CHEN, G., KANDEMIR, M., AND KOLCU, I. 2005. Improving scratch-pad memory reliability through compiler-guided data block duplication. In *Proceedings of the 2005 IEEE/ACM Int. Conf. on Computer-aided design*. ICCAD '05. IEEE Computer Society, Washington, DC, USA, 1002–1005.

LUCENTE, M., HARRIS, C., AND MUIR, R. 1990. Memory system reliability improvement through associative cache redundancy. In *Custom Integrated Circuits Conf., 1990., Proceedings of the IEEE 1990*.

MAKHZAN, M., KHAJEH, A., ELTAWIL, A., AND KURDAHI, F. 2007. Limits on voltage scaling for caches utilizing fault tolerant techniques. In *Computer Design, 2007. ICCD 2007. 25th Int. Conf. on*. 488 –495.

MASTIPURAM, R. AND WEE, E. C. 2004. Soft errors impact on system reliability. In *http://www.edn.com/ article/ CA454636*.

MORRIS, R. J. T. AND TRUSKOWSKI, B. J. 2003. The evolution of storage systems. *IBM Syst. J. 42*.

NASSIF, S. 2001. Modeling and analysis of manufacturing variations. In *Custom Integrated Circuits, 2001, IEEE Conf. on.* 223 –228.

OSCI. 2005. Systemc lrm (ver2.1). *http://www.systemc.org*.

PANDA, P. R., DUTT, N. D., AND NICOLAU, A. 1997. Efficient utilization of scratch-pad memory in embedded processor applications. In *Proceedings of the 1997 European Conf. on Design and Test*. EDTC '97.

PAPIRLA, V. AND CHAKRABARTI, C. 2009. Energy-aware error control coding for flash memories. In *Proceedings of the 46th Annual Design Automation Conf.* DAC '09. ACM, New York, NY, USA, 658–663.

PASRICHA, S. 2002. Tlm of soc with systemc 2.0. *Synopsys User Group Conf. (SNUG)*.

PATTERSON, D. A., GIBSON, G., AND KATZ, R. H. 1988. A case for redundant arrays of inexpensive disks (raid). In *Proceedings of the 1988 ACM SIGMOD Int. Conf. on Management of data*. SIGMOD '88.

PYKA ET AL., R. 2007. Operating system integrated energy aware scratchpad allocation strategies for multiprocess applications. In *Proc.of the 10th Int. workshop on Software & compilers for embedded systems*. SCOPES '07.

RAMASWAMY, S. AND YALAMANCHILI, S. 2007. Improving cache efficiency via resizing + remapping. In *ICCD* (2008-09-22). IEEE, 47–54.

RUCKERBAUER, F. AND GEORGAKOS, G. 2007. Soft error rates in 65nm srams–analysis of new phenomena. In *On-Line Testing Sym., 2007. IOLTS 07. 13th IEEE Int.* 203 –204.

SASAN, A., HOMAYOUN, H., ELTAWIL, A., AND KURDAHI, F. 2009a. A fault tolerant cache architecture for sub 500mv operation: resizable data composer cache (rdc-cache). In *Proceedings of the 2009 Int. Conf. on Compilers, architecture, and synthesis for embedded systems*. CASES '09. 251–260.

SASAN, A., HOMAYOUN, H., ELTAWIL, A., AND KURDAHI, F. 2009b. Process variation aware sram/cache for aggressive voltage-frequency scaling. In *Proceedings of the Conf. on Design, Automation and Test in Europe*. DATE '09.

SHALAN, M. AND MOONEY, V. J. 2000. A dynamic memory management unit for embedded real-time system-on-a-chip. In *Proceedings of the 2000 Int. Conf. on Compilers, architecture, and synthesis for embedded systems*. CASES '00.

SUHENDRA, V., RAGHAVAN, C., AND MITRA, T. 2006. Integrated scratchpad memory optimization and task scheduling for mpsoc architectures. In *Proceedings of the 2006 Int. Conf. on Compilers, architecture and synthesis for embedded systems*. CASES '06. 401–410.

SUHENDRA, V., ROYCHOUDHURY, A., AND MITRA, T. 2008. Scratchpad allocation for concurrent embedded software. In *Proceedings of the 6th IEEE/ACM/IFIP Int. Conf. on Hardware/Software codesign and system synthesis*. CODES+ISSS '08. 37–42.

TAKASE, H., TOMIYAMA, H., AND TAKADA, H. 2010. Partitioning and allocation of scratch-pad memory for priority-based preemptive multi-task systems. In *Proceedings of the Conf. on Design, Automation and Test in Europe*. DATE '10.

THOZIYOOR, S., MURALIMANOHAR, N., AHN, J. H., AND JOUPPI, N. P. 2004. Hp labs cacti v5.3. *CACTI 5.1, TR, http://www.hpl.hp.com/ techreports/2008/ HPL-2008-20.html*.

TILERA. 2010. Tile gx family. *Tilera, http://www.tilera.com/ products/processors/TILE-Gx_Family*.

VERGOS, H. T. AND NIKOLOS, D. 1995. Efficient fault tolerant cache memory design. *Microprocess. Microprogram. 41*, 153–169.

VERMA, M., STEINKE, S., AND MARWEDEL, P. 2003. Data partitioning for maximal scratchpad usage. In *Proceedings of the 2003 Asia and South Pacific Design Automation Conf.* ASP-DAC '03. 77–83.

ZHANG, W. 2004. Enhancing data cache reliability by the addition of a small fully-associative replication cache. In *Proceedings of the 18th annual Int. Conf. on Supercomputing*. ICS '04.

ZHANG, W., GURUMURTHI, S., KANDEMIR, M., AND SIVASUBRAMANIAM, A. 2003. Icr: in-cache replication for enhancing data cache reliability. In *Dependable Systems and Networks, 2003. Proceedings. 2003 Int. Conf. on*.

Received July 2010; revised November 2010; accepted March 2011