ONE STEP MAJORITY LOGIC BASED TRIPPLE ERROR DETECTION DOUBLE ERROR CORRECTION FOR 32-BIT

[1] **Olivia Kopelli,** Pg Scholar, Department of E.C.E, VIKAS GROUP OF INSTITUTIONS, Email id: oliviakopelli@gmail.com.
[2] **Yarram Uma Maheswari,** Assistant professor, Department of E.C.E, VIKAS GROUP OF INSTITUTIONS, Email id: umaecestaff@gmail.com.
[3] **V L N Phani Ponnapalli,** Associate Professor, Department of E.C.E, VIKAS COLLEGE OF ENGINEERING AND TECHNOLOGY, Email id: pvlnphani0454@gmail.com.

Abstract
The reliability of memory subsystem is fast becoming a concern in computer architecture and system design. From on-chip embedded memories in Internet-of-Things (IoT) devices and on-chip caches to off-chip main memories, they have become the limiting factor in reliability of computing systems. This is because they are primarily designed to maximize bit storage density; this makes memories particularly sensitive to manufacturing process variation, environmental operating conditions, and aging-induced wearout. Addressing these concerns is particularly challenging in on-chip caches or embedded memories like scratchpads in IoT devices as additional area, power and latency overheads of reliability techniques in these memories need to be minimized as much as possible. Hence, this dissertation proposes MS-OLS Fault Tolerance in SRAM based scratchpad memories and last level caches. In the first part of the dissertation we propose Difference Set: an approach to deal with known hard faults in software managed scratchpad memories. Difference Set avoids hard faults found during testing by generating a custom-tailored application binary image for each individual chip. During software deployment-time, Difference Set optimally packs small sections of program code and data into fault-free segments of the memory address space and generates a custom linker script for a lazy-linking procedure. The second part proposes two software defined MS-OLS error detection and correction techniques: Software Defined Error Localization Code (SED-DEC) and MS-OLS-ML to recover from soft errors during run time. SED-DEC is mostly for embedded memories and uses novel and inexpensive MS-OLS Error-Localizing Codes (DS-SECs). These require fewer parity bits than single-error-correcting Difference Set codes. Yet our DS-SECs are more powerful than basic single-error-detecting parity: they localize single-bit errors to a specific chunk of a codeword. SED-DEC then heuristically recovers from these localized errors using a small embedded C library that exploits observable side information (SI) about the application's memory contents. MS-OLS-ML is a novel unequal message protection scheme that preferentially provides stronger error protection to certain "special messages". This protection scheme provides Single Error Detection (SED) for all messages and Single Error Correction (SEC) for a subset of special messages. MS-OLS-ML can be used in both last level caches and MS-OLS embedded memories.

1 Introduction

Memories are one of the key bottlenecks in the performance, reliability and energy efficiency of most computing systems. As computing systems have scaled over the decades, the need for memory systems where large amount of data can be stored and retrieved efficiently have also risen rapidly. To achieve this, main memory systems have been scaled for maximum information density. Moore's Law has been the primary driver behind the phenomenal advances in computing

capability of the past several decades. However, with technology scaling having reached the nanoscale era, integrated circuits, especially memory systems, are becoming increasingly sensitive to process variations leading to reliability and yield concerns.

Memories have become the limiting factor in reliability of computing systems [3] because they are primarily designed to maximize bit storage density; this makes memories particularly sensitive to manufacturing process variation, environmental operating conditions, and aging-induced wearout [4, 5]. Unfortunately, errors in computing memories have also increased. In warehouse-scale computers, these errors have become expensive culprits that cause machine crashes, corrupted data, security vulnerabilities, service disruption, and costly repairs and hardware servicing [3, 6]. Google has observed 70000 failures in time (FIT)/Mb in commodity on-chip DRAM memory, with 8% of modules affected per year [3], while Facebook has found that 2.5% of their servers have experienced memory errors per month [7]. The Blue Waters supercomputer had 8.2% of the dual in-line memory modules (DIMMs) (modules that contain multiple RAM chips) encounter an error over the course of a 261 day study [8]. These trends are expected to continue to rise.

Moreover, with IoT devices increasingly becoming part of critical infrastructure and being deployed in failure-intolerant modes (e.g., cars), development of inexpensive fault tolerance schemes for them has become important [9]. Also, with sensing and data-processing being one of the most important use cases for edge devices, these devices are seeing increasing use of large memories. SRAM based scratchpad memories are often the choice of memory architecture used in IoT devices. As demand for higher memory density increases, memory cells are shrunk using advanced technology nodes which in turn makes the memory cells more susceptible to both soft and hard faults. Need for low-power and hence lower operating voltage exacerbates the error rates further. These trends indicate that memory failures are likewise going to be critical for emerging edge/IoT computing devices as well. Low power density is the key to achieving the vision of both exascale computing and the Internet of Things (IoT) [10]. To achieve that, systems need to adopt intelligent power-saving techniques. Memories, both on-chip and off-chip, consume a significant portion of system power. One way to reduce power consumption in on-chip SRAM based memories is to reduce the supply voltage (VDD). However, as shown in Figure 1.1, scaling the VDD down leads to an exponential rise in hard faults in the memory cells [11]. Not only hard faults, the memories also become more susceptible to radiation-induced soft faults at lower voltages, thus degrading yield at low voltage. Moreover, on-chip embedded memories or caches in high performance computing systems are often the largest consumers of chip area. This further increases the likelihood of defects affecting memory rather than logic and process variations with respect to individual memory cells create a significant impact.

To deal with on-chip memory errors due to manufacturing defects designers traditionally include spare rows and columns in the memory arrays [12] and employ large voltage guardbands [13] to ensure reliable operation. Unfortunately, large guardbands limit the energy proportionality of memory. For unpredictable runtime bit flips, the widely used technique to guarantee reliability of storage devices is using information redundancy in the form of Error Correcting Codes (ECC).

ISSN: 0972-2750
Vol-12 Issue-12 No.03: 2022

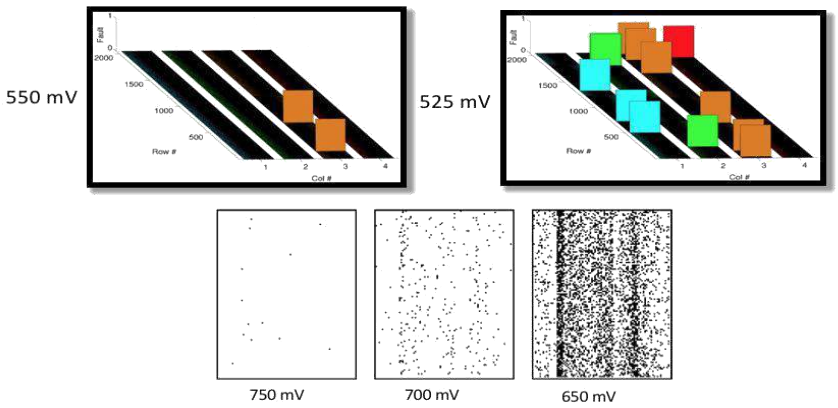

Fig 0: Faults in two SRAM based scratchpad memories at different voltages

In Typical ECCs, extra redundancy bits are added to every row to detect and correct errors. There are additional encoding (while writing data) and decoding (while reading data) procedures required as well. Thus, redundancy in ECC schemes not only incurs area overhead, the encoding and decoding mechanisms also incur additional overheads in terms of latency and energy.

2 Literature Survey

For embedded systems at the edge of the Internet-of-Things (IoT), hardware design is driven by the need for the lowest possible cost and energy consumption, which are both are strongly affected by on-chip memories [14]. Memories consume significant chip area and are particularly susceptible to parameter variations and defects resulting from the manufacturing process [15]. Meanwhile, much of an embedded system's energy is consumed by on-chip SRAM memory, particularly during sleep mode. The embedded systems community has thus increasingly turned to software-managed on-chip memories – also known as scratchpad memories (SPMs) [16] – due to their 40% lower energy as well as latency and area benefits compared to hardware-managed caches [17].

It is challenging to simultaneously achieve low energy, high reliability, and low cost for embedded memory. For example, an effective way to reduce on-chip SRAM power is to reduce the supply voltage [18]. However, this causes cell hard fault rates to rise exponentially [11] and increases susceptibility to radiation-induced soft faults, thus degrading yield at low voltage and increasing cost. Thus, designers traditionally include spare rows and columns in the memory arrays [12] to deal with manufacturing defects and employ large voltage guardbands [13] to ensure reliable opera-tion. Unfortunately, large guardbands limit the energy proportionality of memory, thus reducing battery life for duty-cycled embedded systems [19], a critical consideration for the IoT. Although many low-voltage solutions have been proposed for caches, fewer have addressed this problem for scratchpads and embedded main memory.

Our goal in this work is to improve embedded software-managed memory reliability at minimal cost; we propose a two-step approach. Difference Set first guards applications against known hard faults, which then allows Software-Defined Error-Localizing Codes (SED-DEC) to focus on dealing with unpredictable soft faults. The key idea of this work is to first automatically customize an application binary to individually accommodate each chip's unique hard fault map with no

disruptions to source code, and second, to deal with single-bit soft faults at run-time using novel MS-OLS Error-Localizing Codes (DS-SEC) with a software-defined error handler that knows about the DS-SEC construction and implements a heuristic data recovery policy. The contributions of this chapter are the following.

We present Difference Set, a novel lazy link-time approach that extends the software construction toolchain with new fault-tolerance features for software-managed/scratchpad memories. Difference Set relies on hard fault maps for each software-controlled physical memory region that may be generated during manufacturing test or periodically during run-time using built-in-self-test (BIST).

We detail an algorithm for Difference Set that automatically produces custom hard fault-aware linker scripts for each individual chip. We first compile the embedded program using specific flags to carve up the typical monolithic sections, e.g., .text, .data, stack, heap, etc. Fault-Link then attempts to optimally pack program sections into memory segments that correspond to contiguous regions of non-faulty addresses.

We propose SED-DEC, a hybrid hardware/software technique that allows the system to heuristi-cally recover from unpredictable single-bit soft faults in instruction and data memories, which cannot be handled using Difference Set. SED-DEC relies on side information (SI) about application memory contents, i.e., observable patterns and structure found in both instructions and data. SED-DEC is inspired by our recently-proposed notion of Software-Defined ECC (SDECC) [20].

We describe the novel class of MS-OLS Error-Localizing Codes (DS-SEC) that are used by SED-DEC. DS-SEC codes are stronger than basic single-error-detecting (SED) parity, yet they have lower storage overheads than a single-error-correcting (SEC) Difference Set code. Like SED, DS-SEC codes can detect single-bit errors, yet they can additionally localize them to a chunk of the erroneous codeword. DS-SEC codes can be explicitly designed such that chunks align with meaningful message context, such as the fields of an encoded instruction.

By experimenting with both real and simulated test chips, we find that with no hardware changes, Difference Set enables applications to run correctly on embedded memories using a min-VDD that can be lowered by up to 440 mV. After Difference Set has avoided hard faults (that may include defects as well as voltage-induced faults), our SED-DEC technique recovers from up to 90% of random single-bit soft faults in 32-bit data memory words and up to 70% of errors in instruction memory using a 3-bit DS-SEC code (9.375% storage overhead). SED-DEC can even be used to recover up to 70% of errors using a basic SED parity code (3.125% storage overhead). In contrast, a full Difference Set SEC code incurs a storage overhead of 18.75%. Our combined Difference Set+SED-DEC approach could thus enable more reliable IoT devices while significantly reducing cost and run-time energy.

To the best of our knowledge, this is the first work to both (i) customize an application binary on a per-chip basis by lazily linking at software deployment-time to accommodate the unique patterns of hard faults in embedded scratchpad memories, and (ii) use error-localizing codes with software-defined recovery to cope with random bit flips at run-time.

3 Exisitng Scheme

There is an abundance of prior work on fault-tolerant and/or low-voltage caches. Examples include PADded Cache [47], Gated-VDD [48], Process-Tolerant Cache [49], Variation-Aware Caches [50], Bit Fix/Word Disable [51], ZerehCache [52], Archipelago [53], FFT-Cache [54], VS-ECC [55], Correctable Parity Protected Cache (CPPC) [56], FLAIR [57], Macho [58], DPCS [59], DARCA [60], and others (see related surveys by Mittal [61, 4]). These fault-tolerant cache techniques tolerate hard faults/save energy by sacrificing capacity or remapping physical data locations. This affects the software-visible memory address space and hence they cannot be readily applied to SPMs.

Although they are cache-specific, some of the above techniques can be roughly compared with Difference Set in terms of min-VDD. For instance, DPCS [59] achieves a similar min-VDD to Difference Set of around 600 mV, while FLAIR [57] achieves a lower min-VDD (485 mV). We emphasize that the above techniques cannot be applied to SPMs and are therefore not a valid comparison.

Similar to SED-DEC, CPPC [56] can recover random soft faults using SED parity. However, CPPC requires additional hardware bookkeeping mechanisms that are in the critical path whenever data is added, modified, or removed from the cache (and again, their method is not applicable to SPMs).

3.1 Fault-Tolerant Scratchpads

The community has proposed various methods for tolerating variability and faults in SPMs that relate closely to this work. Traditional fault avoidance techniques like dynamic bit-steering and strong ECC codes are too costly for small embedded memories. Meanwhile, spare rows and columns cannot scale to handle many faults that arise from deep voltage scaling. E-RoC is a SPM fault-tolerance scheme that aims to dynamically allocate scratchpad space to different applications on a multi-core embedded SoC using a virtual memory approach. However, it requires extensive hardware and run-time support. Several works propose to use OS-based virtual memory to directly manage memory variations and/or hard faults, but they are not feasible in low-cost IoT devices that lack support for virtual memory; nor do they guarantee avoidance of known hard faults at software deployment time. Others have proposed to add small fault-tolerant buffers that assist SPM checkpoint/restore re-compute corrupted data upon detection build radiation-tolerant SPMs using hybrid non-volatile memory [66] and duplicate data storage to guard against soft errors these are each orthogonal to this work.

There are several other prior works that relate closely to SED-DEC, although ours is the first to propose heuristic recovery that lies in the regime between SED and SEC capabilities. Farbeh et al. [68] propose to recover from soft faults in instruction memory by leveraging basic SED parity combined with a software recovery handler that leverages duplicated instructions in memory. On the other hand, our approach does not add any storage overhead to recover from errors (although ours is heuristic). Volpato et al. [69] proposed a post-compilation binary patching approach to improve energy efficiency in SPMs that closely resembles the Difference Set procedure. However, that work did not deal with faults in the SPMs uses SED parity to dynamically recompute critical data that that is affected by single-bit soft faults. SED-DEC completely subsumes that approach:

the embedded SED-DEC library can heuristically recover data, recompute it if possible, or opt to panic according to the application's needs.

Unlike all known prior work, our combined Difference Set+SED-DEC approach can simultaneously deal with both hard and soft SPM faults with minimal hardware changes compared to existing IoT systems. Our low-cost approach can be used today with off-the-shelf microcontrollers (minor changes are needed to implement DS-SEC codes, however), and can improve yield and min-VDD.

4 Proposed System

As demand and size of on-chip caches is increasing rapidly and the physical dimension and noise margins are decreasing, reliability of caches is increasingly becoming an important issue. As given in the vulnerability of SRAM caches to soft errors grows with increase in size. Also with reduction in physical dimensions of these devices, the critical charge required to flip the content of a cell due to a particle strike decreases. As a result, the soft error rate is higher for large capacity caches. The widely used technique to guarantee reliability of storage devices is using information redundancy in the form of Error Correcting Codes (ECC). In typical ECCs, extra redundancy bits are added to every row to detect and correct errors. There are additional encoding (while writing data) and decoding (while reading data) procedures required as well. Thus ECCs come with encoding and decoding mechanisms that incur additional overheads in terms of latency and energy. Both these overheads are critical for caches and hence, ECC protection was not widely used in caches till recently. However, due to the increased reliability concerns of large capacity caches and processor performance degradation due to occurrence of errors, cache protection using ECC schemes is becoming increasingly popular. Nevertheless, these additional area, power and latency overheads need to be minimized in caches as much as possible.

We present MS-OLS-ML: a novel unequal message protection scheme for last level caches that preferentially provides stronger error protection to certain "special messages". As the name suggests, this coding scheme requires one extra bit above a simple parity Single Error Detection (SED) code while providing SED for all messages and Single Error Correction (SEC) for a subset of messages. Thus, it is stronger than just basic SED parity and has 9% lower storage overhead than a traditional Single Error Correcting, Double Error Detecting (SECDED) code. Error detection circuitry often lies on the critical path and is generally more critical than error correction circuitry as error occurrences are rare even with an increasing soft error rate. Our coding scheme has a much simpler error detection circuitry that incurs lower energy and latency costs than the traditional SECDED code. Thus, MS-OLS-ML is a MS-OLS ECC code that is ideal for large capacity last level caches. We also propose a memory speculation procedure that can be generally applied to any ECC protected cache to hide the decoding latency while reading messages when there are no errors. Error-correcting codes (ECCs) increase the resiliency of communication and storage systems by adding redundant bits (or symbols, but in this work we focus on the binary regime). A code C can be thought of as an injective mapping of messages of length k to codewords of length n. Let r be the number of redundant bits, i.e., r = n k. A binary code is considered linear if the sum of any two codewords in C is also a codeword in C. A linear block code is described by either its (kxn) generator matrix G or its (rxn) parity-check matrix H, with the relation $GH^T = 0$. A particular message m is encoded to its corresponding codeword c by multiplying it with the generator matrix as follows: mG = c. Each row of H is a parity-check equation that all codewords

must suffice, thus $Hc^T = 0$. We define the received vector at the output of the channel as y = c + e, in which e is the error-vector representing which bits have been flipped. The receiver calculates the syndrome, $s = Hy^T$, and if s 6= 0, then it is known that the received vector is not a valid codeword. At this point, the decoder can either attempt to determine the most likely originally transmitted codeword or it can simply raise a flag that an error was detected (depending on the system goals and design). We say a code is systematic if a message is directly embedded in the codeword, i.e., each message bit is equal to a specific codeword bit.

A useful parameter of a linear code is its minimum distance, d_{min}, which is the minimum Difference Set distance between any two (non-identical) codewords. Additionally, since a linear code must include the 0 codeword, the minimum distance of a linear code is simply the minimum weight.

A linear code guarantees correction of up to $t = b\frac{1}{2}(d_{min} 1)c$ bit-errors, or detection of up to (d_{min} 1) bit-errors (without any correction guarantees). For even values of d_{min}, a linear code simultaneously guarantees correction of up to t bit-errors and detection of up to (t + 1) bit-errors. Further explanation of the fundamental properties of codes can be found in classic textbooks [79, 80].

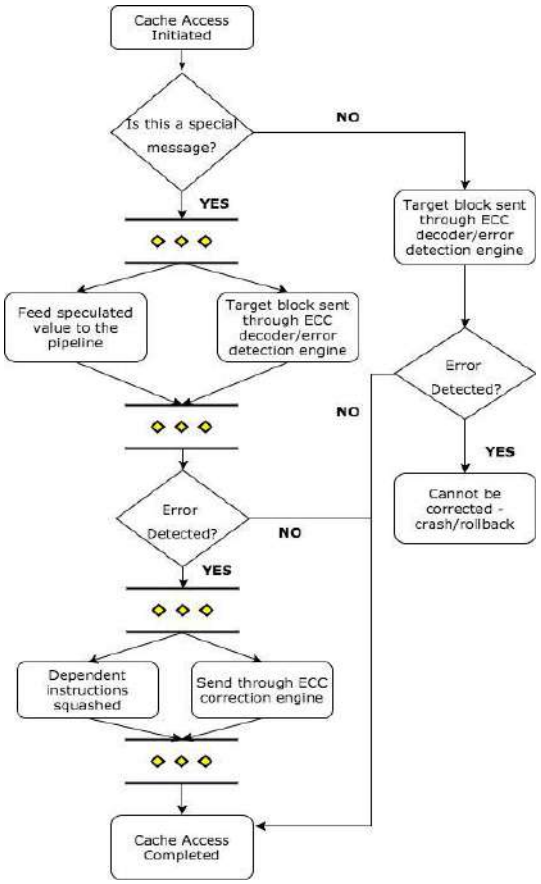

Fig 2: Flow of read operation in cache with memory speculation and MS-OLS-ML protection schemes

Results

In this section we discuss the performance results obtained from the Gem5 simulations In both the evaluations, performance of the system with MS-OLS-ML was compared against that with

SECDED. The evaluation was further split into iso-area and iso-capacity as explained in For both the core configurations, the observations for the iso-area case are almost similar. With memory speculation it is seen that with additional memory capacity for iso-area, the system with MS-OLS-ML has upto 4% better performance (lower execution time) than the one with SECDED. This improvement in performance happens in spite of the additional one cycle latency incurred on non special messages in the case of MS-OLS-ML. The applications showing higher performance benefits are mostly memory intensive. Hence, additional cache capacity with MS-OLS-ML reduces overall miss rate to an extent such that the slight increase in average LLC hit time gets offset. For most of these applications, this performance gap widens as the LLC size increases for Processor-II. The applications showing roughly similar performances on both the systems are the ones which already have a considerably lower LLC miss rate. As a result, increase in LLC capacity due to MS-OLS-ML doesn't lead to a significant improvement in performance. The same evaluation was also done for the case where there is no memory speculation, i.e., both MS-OLS-ML and SECDED protected caches have additional hit latency of one cycle for all read operations. The results show that with the exact same hit latency, MS-OLS-ML has upto 7% lower execution time than SECDED due to additional memory capacity.

A more significant result is the iso-capacity case with memory speculation. It is seen that even with additional one cycle latency for non special messages in MS-OLS-ML, the performance of the system with MS-OLS-ML is at par with that of SECDED. This means that by using our MS-OLS error correction scheme, we manage to save about 5-9% last level cache area (excluding decoder and peripheral circuit area) with negligible hit in performance. Since the LLCs constitute more than 30% of the processor chip area, the cache area savings translate to a considerable amount of reduction in the chip size. This additional area benefit can either be utilized to make an overall smaller sized chip or it can be used to pack in more compute tiles to increase the overall performance of the system.

The ISO-capacity results also imply that MS-OLS-ML can be used in SRAM based scratchpad memories used in embedded systems at the edge of the Internet-of-Things (IoT) where hardware design is driven by the need for low area, cost and energy consumption. Since MS-OLS-ML helps in reducing area (in turn reducing SRAM leakage energy) and also has lower error detection energy, it provides a better protection mechanism than SECDED in such devices.

Fig 3 Simulation Result for MS-OLS-MLD

ISSN: 0972-2750
Vol-12 Issue-12 No.03: 2022

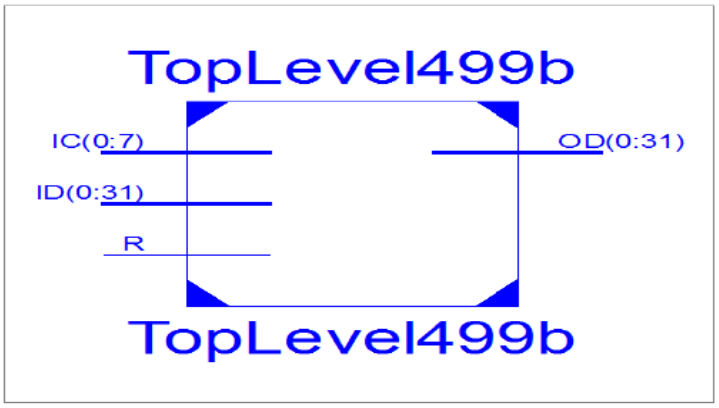

Fig 4: Top Level

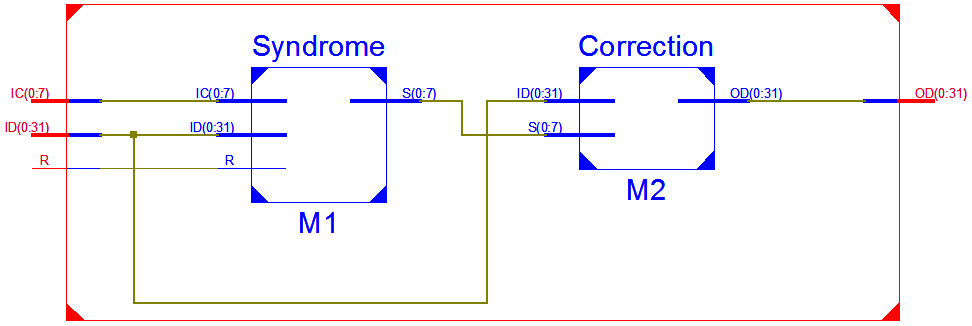

Fig 5: Internal architecture

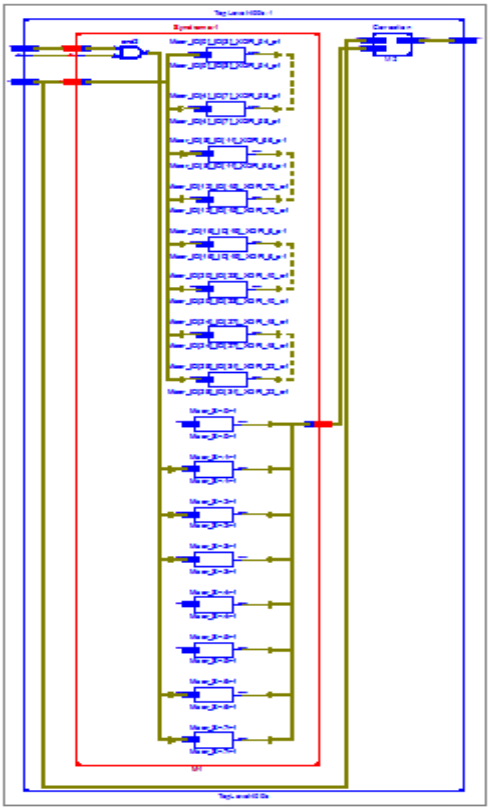

Fig 6 Complete internal architecture

Simulation results of encoder are shown in fig.8 Here IN(31:0) are the input and OUT(38:0) is the output. . These are synthesized and simulated using Xilinx ISE 14.7 tool for vertex family device and simulation results as well as synthesis reports are presented.

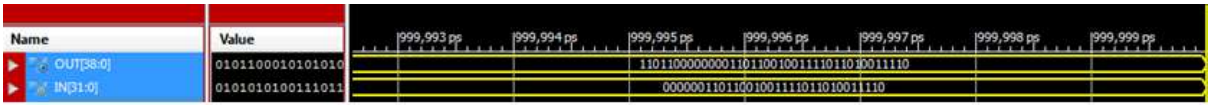

Fig 7: Simulation Results of encoder

The RTL view of the encoder is shown in below fig.5.4(a) ,5.4(b). It shows the internal blocks and connections of the architecture.

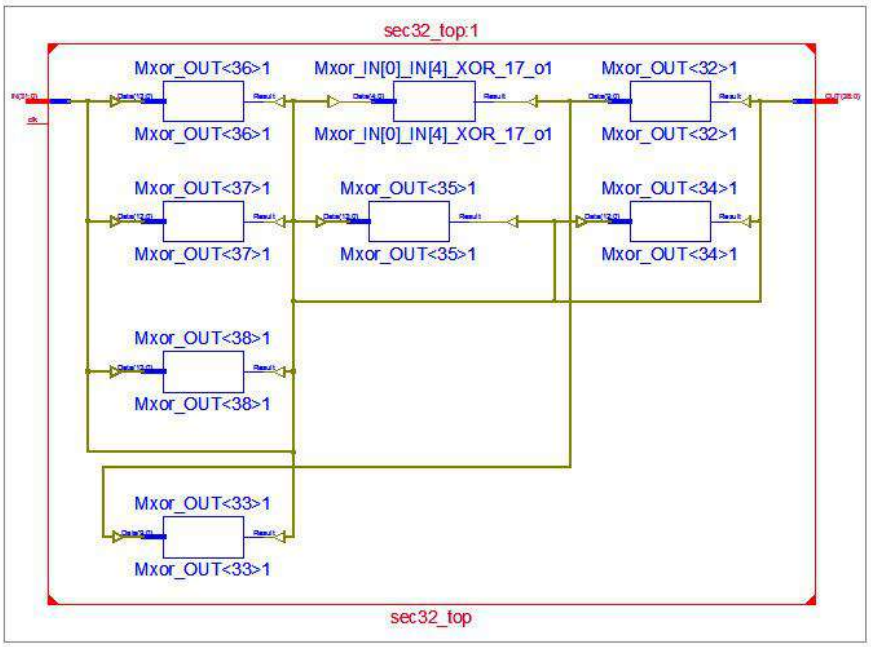

Fig 8 Complete internal architecture

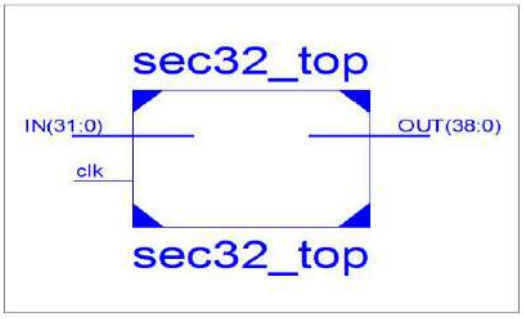

Fig 9 RTL Schematic

Conclusion

A series of techniques were proposed to cope with hardware variability and errors in on-chip memories. With increase in size of on-chip memories and decrease in physical dimensions of cells, memory reliability is becoming a growing concern. The challenge with these memories is that the fault tolerance techniques need to effective but with minimal overhead. SED-DEC provided a holistic virtualization-free fault tolerance methodology to deal with hard and soft faults in software

managed embedded memories in IoT devices. Hardware design in most of these IoT devices is driven by the need for low cost and low power. One way to reduce power consumption is to lower the supply voltage. But as the VDD is lowered, some of the weak SRAM memory cells begin to fail. Hence, low cost protection against hard faults in memory is required if these devices have to be run at low voltage. Difference Set does exactly that with almost no hardware overhead. In software managed memories, data placement in memory is orchestrated by the software. Thus, application programmers, with the help of tools like compiler and linker, explicitly partition data into physical memory regions that are distinct in the address space. Difference Set utilizes exactly that property of software managed memories and makes loading application in faulty memory plausible. It takes a pre-compiled binary of an application and links it to the memory in such a way that the application would not access the bad locations. Thus, the application is compiled once but the final linked binary image is unique for every chip. SED-DEC, on the other hand, helps to recover from unpredictable single bit flips in the memory that occur during runtime. It helps to localize the error to a smaller chunk in a 32/64-bit message and then tries to heuristically recover from it using software defined policies that leverage on the available side information about memory contents to choose the most likely candidate codeword. Overall, SED-DEC together opportunistically copes with memory errors in low-cost IoT devices and helps in improving the longevity of these devices.

REFERENCES

[1] F. J. Aichelmann, “Fault-Tolerant Design Techniques for Semiconductor Memory Applica-tions,” IBM Journal of Research and Development, vol. 28 , no. 2, pp. 177–183, 1984.

[2] P. P. Shirvani and E. J. McCluskey, “PADded Cache: A New Fault-Tolerance Technique for Cache Memories,” in Proceedings of the VLSI Test Symposium, 1999.

[3] A Circuit Technique to Reduce Leakage in Deep-Submicron Cache Memories,” in Proceedings of the IEEE International Symposium on Low Power Electronicsand Design (ISLPED), 2000.

[4] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, “MiBench: A Free, Commercially Representative Embedded Benchmark Suite,” in Proceedings of the IEEE International Workshop on Workload Characterization (IWWC), 2001.

[5] S. Hamdioui, A. J. van de Goor, and M. Rodgers, “March SS: A Test for All Static Simple RAM Faults,” in International Workshop on Memory Technology, Design, and Testing (MTDT), 2002.

[6] A. Agarwal, B. C. Paul, H. Mahmoodi, A. Datta, and K. Roy, “A Process- Tolerant Cache Architecture for Improved Yield in Nanoscale Technologies,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems , vol. 13, no. pp. 27–38, 2005.

[7] F. Li, G. Chen, M. Kandemir, and I. Kolcu, “Improving Scratch-Pad Memory Reliability Through Compiler-Guided Data Block Duplication,” in Proceedings of the IEEE/ACM Inter-national Conference on Computer-Aided Design (ICCAD), 2005.

[8] M. Mutyam and V. Narayanan, “Working with Process Variation Aware Caches,” in Design, Automation, and Test in Europe (DATE), 2007.

[9] C. Wilkerson, H. Gao, A. R. Alameldeen, Z. Chishti, M. Khellah, and S.-L. Lu,“Trading off Cache Capacity for Reliability to Enable Low Voltage Operation,” in Proceedings of the ACM/IEEE International Symposium on Computer Architecture (ISCA), 2008.

[10] A. Ansari, S. Gupta, S. Feng, and S. Mahlke, "ZerehCache: Armoring Cache Architectures in High Defect Density Technologies," in Proceedings of the ACM/IEEE International Symposium on Microarchitecture (MICRO), 2009.

[11] D. P. Volpato, A. K. Mendonca, L. C. dos Santos, and J. L. Guntzel,¨ "A Post- Compiling Approach that Exploits Code Granularity in Scratchpads to Improve Energy Efficiency," in Proceedings of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI), pp. 127–132, 2010.

[12] L. A. D. Bathen and N. D. Dutt, "E-RoC: Embedded RAIDs- on-Chip for Low Power Distributed Dynamically Managed Reliable Memories," in Design, Automation, and Test in Europe (DATE), 2011.

[13] A. Ansari, S. Feng, S. Gupta, and S. Mahlke, "Archipelago: A Polymorphic Cache Design for Enabling Robust Near-Threshold Operation," in Proceedings of the IEEE International Symposium on High Performance Computer Architecture (HPCA), 2011.

[14] A. BanaiyanMofrad, H. Homayoun, and N. Dutt, "FFT-Cache : A Flexible Fault-Tolerant Cache Architecture for Ultra Low Voltage Operation," in Proceedings of the ACM/IEEE International Conference on Compilers, Architectures and Synthesis for Embedded Systems (CASES), 2011.

[15] A. R. Alameldeen, I. Wagner, Z. Chishti, W. Wu, C. Wilkerson, and S.-L. Lu, "Energy-Efficient Cache Design Using Variable-Strength Error-Correcting codes in Proceedings of the ACM/IEEE International Symposium on Computer Architecture (ISCA), 2011.

[16] M. Manoochehri, M. Annavaram, and M. Dubois, "CPPC: Correctable Parity Protected Cache," in Proceedings of the ACM/IEEE International Symposium on Computer Architecture (ISCA), 2011.

[17] H. Farbeh, M. Fazeli, F. Khosravi, and S. G. Miremadi, "Memory Mapped SPM: Protecting Instruction Scratchpad Memory in Embedded Systems against Soft Errors," in Proceedings of the European Dependable Computing Conference (EDCC), 2012.

[18] L. A. D. Bathen, N. D. Dutt, A. Nicolau, and P. Gupta, "VaMV: Variability-Aware Memory Virtualization," in Design, Automation, and Test in Europe (DATE), 2012.

[19] A. Sampson, J. Nelson, K. Strauss, and L. Ceze, "Approximate Storage in Solid-State Memories," in Proceedings of the IEEE/ACM International Symposium on Microarchitecture (MICRO), 2013.

[20] A Fault-Tolerant ScratchPad Memory," in Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 2013.