(Invited) Cross-Layer Resilience: Challenges, Insights, and the Road Ahead

Eric Cheng¹, Daniel-Mueller-Gritschneider², Jacob Abraham³, Pradip Bose⁴, Alper Buyuktosunoglu⁴, Deming Chen⁵,
Hyungmin Cho⁶, Yanjing Li⁷, Uzair Sharif², Kevin Skadron⁸, Mircea Stan⁸, Ulf Schlichtmann², Subhasish Mitra¹

¹Stanford University, ²Technical University of Munich, ³University of Texas at Austin, ⁴IBM Research, ⁵University of Illinois at Urbana-Champaign, ⁶Hongik University, ⁷University of Chicago, ⁸University of Virginia

ABSTRACT

Resilience to errors in the underlying hardware is a key design objective for a large class of computing systems, from embedded systems all the way to the cloud. Sources of hardware errors include radiation, circuit aging, variability induced by manufacturing and operating conditions, manufacturing test escapes, and early-life failures. Many publications have suggested that cross-layer resilience, where multiple error resilience techniques from different layers of the system stack cooperate to achieve cost-effective resilience, is essential for designing cost-effective resilient digital systems. This paper presents a comprehensive overview of cross-layer resilience by addressing fundamental cross-layer resilience questions, by summarizing insights derived from recent advances in cross-layer resilience research, and by discussing future cross-layer resilience challenges.

CCS CONCEPTS

• General and reference → Reliability; • Hardware → Fault tolerance; • Computer systems organization → Reliability

KEYWORDS

Cross-layer resilience, fault tolerance, reliability

Q1. What is cross-layer resilience?

Digital systems can be subject to *errors* in the underlying hardware. Errors can prevent correct system operation causing system malfunction. Sources of such *hardware errors* include radiation-induced transient errors (*soft errors*), circuit aging, variability induced by manufacturing and operating conditions (e.g., *voltage noise*), manufacturing test escapes, and early-life failures (*ELF*). At the same time, there is explosive growth in our dependency on digital systems – the impacts of their malfunctions continue to increase as systems become more complex, interconnected and pervasive (e.g., automotive safety [ISO 11] or cyber-physical systems in harsh environments [DIN 16]). Ensuring *resilient* operation, i.e., the system operates correctly despite hardware errors, is crucial for a large class of systems, from embedded systems all the way to the cloud [Mitra 14].

How do we design **cost-effective** resilient digital systems? Here *cost* refers to energy, execution time, power, or area costs. Many

publications (e.g., [Borkar 05, Cappello 14, Carter 10, DeHon 10, Gupta 14, Henkel 14, Pedram 12]) have suggested that *cross-layer resilience*, where error resilience techniques from various layers of the system stack cooperate to achieve cost-effective resilience, is essential to overcome this challenge.

Q2. Literature on resilient system design dates back decades. What's new about cross-layer resilience?

There are numerous publications on resilience techniques (e.g., fault-tolerant computing) over decades. Some of these techniques span multiple abstraction layers (e.g., watchdog processors and control-flow checking [Lu 82, Mahmood 88], Algorithm-Based Fault-Tolerance [Huang 84]) while some don't (e.g., Software Implemented Hardware Fault Tolerance [Lovelette 02]). Resilience implementations in commercial systems (e.g., [Sinhavoy 11]) often span multiple layers of the system stack. However, until recently (i.e., before CLEAR [Cheng 16, 17, 18a, 18b, Cho 17], details in Question 3), there existed little holistic understanding of cross-layer resilience. Several preconceptions existed, and it was difficult to provide quantitative answers to important cross-layer resilience questions such as:

- (a) Is cross-layer resilience the best approach for achieving a given resilience target at low cost?
- (b) Are all cross-layer solutions equally cost-effective? If not, which ones are the best and why?
- (c) How do cross-layer choices change depending on application-level energy, latency, and area constraints?
- (d) How can one create a cross-layer resilience solution that is cost-effective across a wide variety of application workloads?
- (e) Are there general guidelines for new error resilience techniques to be cost-effective?

Q3. What is CLEAR?

CLEAR, an acronym for Cross-Layer Exploration for Architecting Resilience, is a first of its kind software framework for comprehensive exploration of cross-layer resilience. CLEAR [Cheng 16, 17, 18a, 18b, Cho 17] starts with a representative set of existing resilience techniques spanning circuit, logic, architecture, software, and application layers. It then analyzes combinations of these techniques (*cross-layer combinations*), derives cost-effective resilience solutions (to protect a given design from a variety of error sources such that error resilience targets are met at minimal costs), quantifies their cost-benefit tradeoffs, and explains the causes for their cost-effectiveness. The in-depth analysis of CLEAR offers insights into how to create cost-effective resilience techniques.

CLEAR has been applied to a wide range of diverse hardware designs (general-purpose processor cores, hardware accelerators, uncore components such as cache controllers, memory controllers and I/O controllers) used in System-on-Chips (*SoCs*). CLEAR results have also been validated for industrial designs.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

DAC'19, June, 2019, Las Vegas, USA

© 2018 Copyright held by the owner/author(s). 978-1-4503-0000-0/18/06...\$15.00

DOI: 10.1145/3316781.3323474

For example, CLEAR thoroughly explores the soft error resilience design space (586 cross-layer combinations) and derives resilience solutions that achieve a 50-fold improvement in silent data corruption (SDC) rate (incorrect application outputs with no error indication) for out-of-order processor cores at 2% energy cost and no speed impact (by combining application-guided selective circuit-level hardening using LEAP-DICE [Lee 10, Lilja 13], logic parity, and micro-architecture recovery using reorder buffer). Similar explorations demonstrate a 50-fold improvement in SDC rate at 1.6% and 5.7% energy costs for uncore components and hardware accelerators, respectively. The voltage noise design space is explored to derive a solution for in-order processor cores at 2.1% energy cost.

Q4. What are the key components of CLEAR?

The CLEAR framework encompasses the entire design flow end-to-end: from comprehensive and thorough analysis of various resilience techniques (and their combinations) to design heuristics for creating cost-effective cross-layer combinations, all the way to layout-level implementations that capture energy, power, area, and execution time costs. Through extensive exploration, CLEAR thus provides insights into how to create cost-effective cross-layer solutions or why particular combinations are more cost-effective than others, and also helps dispel many myths surrounding cross-layer resilience. Key components of CLEAR include:

Extensive resilience library that covers a wide variety of resilience (error correction, error detection, error recovery, proactive circuit failure prediction, throttling) techniques at the circuit, logic, architecture, software, and application levels (e.g., 586 cross-layer combinations for soft error resilience, 244 for voltage noise).

Accurate reliability analysis to estimate the application-level impact of hardware errors (Question 5). CLEAR employs numerous detailed error injections using FPGA emulation, the Stampede supercomputer, and mixed-mode simulation techniques for this purpose.

Execution time and power estimation using FPGA emulation, RTL simulation, and industrial sign-off tools to accurately capture system-level behaviors while running application benchmarks.

Detailed physical design to capture overheads of resilience techniques. Without such careful analysis, publications often report unrealistic projections that overlook hidden costs (e.g., error signal routing or hardware recovery costs).

Q5. How do you estimate the application-level impact of hardware errors?

Numerous publications quantify the circuit-level susceptibility of digital designs to soft errors, voltage noise or circuit aging before (or up to the point when) errors get captured in flip-flops. This question addresses a different challenge: once errors get captured in flip-flops, how does one estimate their application-level impact? Error injection is widely used for this purpose.

Most error injections target soft errors (for reasons explained later in this section). Many such existing error injection techniques suffer from drastic accuracy challenges (details in [Cho 13, Mirkhani 14]). For example, our CLEAR results show ([Cho 13], also confirmed by [Mueller-Gritschneider 18a]) that existing high-level soft error injections (that inject single-bit errors randomly into software-visible registers or program variables) can be highly inaccurate (e.g., 0.07-fold underestimation to 45-fold

overestimation of application-level soft error vulnerability). Such inaccuracies can result in overly pessimistic design decisions with inefficient overprotection, or optimistic design choices (as discussed in Question 6). While flip-flop-level soft error injections are generally accurate (and CLEAR uses millions of such flip-flop-level injections), they can be extremely slow, especially for uncore components [Cho 17]. Careful mixed-mode soft error simulations (e.g., [Cho 17, Mueller-Gritschneider 18a]) can achieve high accuracy and high throughput (e.g., 20,000-fold speedup compared to RTL-only simulation for uncore components [Cho 17], used by CLEAR).

Accurate error injection for voltage noise or circuit aging is difficult because it requires detailed application-level simulations considering circuit-level timing (and their variations) over many clock cycles. While bit flip models have been suggested for timing errors, their efficacy is unclear. Hence, CLEAR analyzes voltage noise at the circuit level, and does not propagate the effects of errors to the application level. Instead, CLEAR uses circuit-level proactive circuit failure prediction/error detection and architecture-level throttling/error recovery techniques. Thus, CLEAR derives a cost-effective voltage noise resilience solution (Questions 3 and 6) while avoiding simulations that propagate voltage noise-induced errors to the application level.

An inaccurate error injection technique can be useful as long as it is effective in deriving cost-effective resilience solutions. One such example exists in the digital testing literature. Stuck-at faults are inaccurate in modeling manufacturing defects, but are highly effective as test metrics that drive automatic test generation and design for testability techniques [McCluskey 00]. Future research must explore this aspect of error models for cross-layer resilience.

Q6. What key insights have been derived from CLEAR?

1. *Circuit-level techniques are crucial* for cost-effective error resilience (e.g., LEAP-DICE error correction [Lee10, Lilja 13] for soft errors or circuit failure prediction using critical path monitors or CPM [Lefurgy 11] for voltage noise). For example, application-guided selective LEAP-DICE insertion achieves a 50-fold improvement in soft error-induced SDC rate for out-of-order processor cores at 3% energy cost (no speed impact). For accelerators, similar improvement is achieved at 7% energy cost.

2. *Application-level guidance is essential* (e.g., to identify vulnerable design blocks) for cost-effective resilience. However, some application-level knowledge may be unavailable at design time. CLEAR demonstrates ways to overcome this challenge (Question 7), but more research is required moving forward.

3. *Existing architecture- and software-level resilience techniques are generally expensive* (16-110% energy costs) and their low-cost variants provide too little resilience (e.g., less than 2-fold SDC rate improvement). Hence, circuit-level techniques are crucial (Insight 1 above). In some cases, application-level techniques can be cost-effective, e.g., output timing monitoring using a watchdog timer reduces unsafe application behaviors 63-fold for embedded control [Mueller-Gritschneider 18a].

4. *Some of the previously-published resilience techniques, in particular those implemented at the architecture and software layers, suffer from incorrect evaluation.* CLEAR uses accurate evaluation methods to correct some of these incorrect conclusions and optimistic projections from the past. For example, injecting random errors into software-visible registers (Question 5) might lead to an optimistic estimation of 19-fold (vs. actual 2-fold) SDC

rate improvement (for soft errors) using certain software-level resilience techniques. Optimistic projections may also result from overlooking hidden costs in physical designs (Question 4).

5. *Cost-effective protection from multiple error sources (e.g., soft errors and voltage noise) is achieved by combining resilience techniques that specifically target each error source.* For example, an in-order processor core can achieve combined soft error and voltage noise resilience at 8.2% energy cost vs. 28% or more using resilience techniques that are agnostic to error sources.

6. *CLEAR derives highly effective and low-cost resilience techniques by combining existing techniques.* Hence, new resilience techniques must be extremely efficient in terms of cost vs. degree of resilience achieved to be competitive.

Q7. How can you incorporate application-level guidance for cost-effective error resilience?

As stated in Question 6, application-level guidance is essential for cost-effective error resilience. For general workloads, it is well-known that not all errors cause incorrect application-level behaviors (e.g., SDCs) – this makes circuit-level selective protection possible. For embedded systems (e.g., robot control), even application-level SDCs may not cause unsafe behaviors (e.g., only 25% of SDCs cause unsafe robot trajectories [Mueller-Gritschneider 18b]). Many data analysis and machine learning workloads exhibit *algorithmic* (algorithms inherently tolerant) and *cognitive* (incorrect results may be “acceptable” to human users) resilience [Chandramoorthy 19, Cho 12, Whatmough 18 and others]. These applications tolerate data errors but control-flow errors can be catastrophic. This observation creates opportunities for asymmetric resilience architectures (e.g., ERSA [Cho 12]).

However, not all applications may be known at design time. For general-purpose processor cores, CLEAR overcomes this challenge using specially-designed circuit-level techniques that augment resilience at 1% additional energy cost. This approach assumes that “unknown” applications are “similar” to those known at design time. Hardware accelerators exhibit similar data input dependence. While the workload (e.g., software control tasks) and target mission profiles (e.g., typical sensor inputs) may be available at design time for some embedded systems, a “good” understanding of the trustworthiness of such information is still required. For truly robust design practices, pre-silicon design should explore corner-case “stress” workloads (e.g., [Bertran 12, Mirkhani 15]), across suitable data inputs. Thus, to ensure target in-field coverage, further CLEAR refinements could incur higher energy costs. Future research must understand related bounds.

For neural network accelerators, considerations of cognitive resilience extend beyond the realm of data-dependent uncertainty. Inference accuracy in the field depends on the representativeness of the in-factory training data. Deviations in field data away from training data characteristics could result in significant inference accuracy loss. System resilience under this class of data anomalies is crucial for safety-critical applications (e.g., [Vega 18]).

Q8. How do you implement on-line self-test and diagnostics?

On-line self-test and diagnostics, that enable a system to test itself concurrently during normal operation without any downtime visible to the end-user, may be used for proactive circuit failure prediction (e.g., for ELF using special ELF signatures [Kim 10, 13, Sauer 13] or circuit aging [Agarwal 07, Baba 09, Tschanz 09]), for detecting *hard failures* (e.g., ELF, circuit aging, manufacturing

defects), or for diagnosis [Mitra 14]. Careful calibration is required to account for temperature and voltage variations when delay tests are applied during system operation.

CASP, an acronym for *C*oncurrent *A*utonomous chip self-test using *S*tored test *P*atterns, is an elegantly simple yet highly effective technique for thorough on-line self-test and diagnostics [Li 08, 10, 13a]. CASP can be applied to complex SoCs containing processor cores, accelerators, and uncore components.

A key aspect of CASP is cross-layer orchestration (physical design, architecture, system software). For processor cores, software-orchestration (virtualization [Inoue 08], CASP-aware operating system scheduling [Li 09]) achieves 0.3% execution time impact. For uncore components, CASP incurs 1% area and power costs, and 3% execution time impact for the OpenSPARC T2 SoC [OpenSPARC]. With additional area, CASP can achieve ultra-short chip test times (e.g., 2ms for OpenSPARC T2) that can benefit manufacturing testing as well [Li 13a].

CASP also enables low-cost on-line diagnosis by matching block-level diagnosis granularity with *self-repair granularity*, i.e., hardware blocks replaced/bypassed if failures occur inside these blocks (see self-repair discussion in Question 9).

Q9. What are cost-effective ways of implementing error recovery and self-repair?

For soft errors, architecture-level recovery techniques alone can be very expensive. Hence, CLEAR derives cost-effective solutions by combining circuit-level LEAP-DICE error correction with various flavors of processor recovery techniques and Quick Replay Recovery [Cho 17] for uncore components. For voltage noise, proactive circuit failure prediction using CPMs followed by error prevention (e.g., using architecture-level instruction throttling) is crucial. For circuit aging, on-line self-tuning has been extensively explored (e.g., [Mintarno 11 and many others]).

For (permanent) hard failures, upon circuit failure prediction or hard failure detection, self-repair techniques are essential to keep the system functioning correctly. Unlike on-chip memories and processor cores, naïve redundancy-based sparing techniques for uncore components impose significant (e.g., 20%) chip-level area costs. New self-repair techniques for uncore components [Li 13b] achieve high (98%) *self-repair coverage* (i.e., the probability that the system functions correctly in the presence of any single faulty uncore component) at only 7.5%, 3%, and 0.3%-5% chip-level area, power, and execution time impact, respectively. These techniques do not introduce execution time impact in fault-free systems and can also be used to improve manufacturing yield.

Q10. What cross-layer resilience research opportunities exist for future systems?

Two topics, not covered earlier in this paper, are: resilience in the beyond-silicon era, and resilience-security interactions.

Resilience in the beyond-silicon era: For coming generations of abundant-data applications, transformative NanoSystems such as *N3XT* [Aly 15, 19] are essential. *N3XT* creates new computing system architectures by leveraging new device nanotechnologies (e.g., carbon nanotube field-effect transistors, high-density non-volatile resistive memory) and their ultra-dense (e.g., monolithic) 3D integration with fine-grained connectivity for computation immersed in memory. Such NanoSystems create new cross-layer resilience research opportunities. Examples include ways to address inherent imperfections and variations in device

nanotechnologies, write failures (e.g., endurance) in non-volatile memories, variations associated with multi-valued storage inside non-volatile memory and their application-level impact, and fine-grained self-repair to overcome potential yield challenges of monolithic 3D integration [Aly 19, Grossi 19, Le 19, Wu 18, 19].

Resilience-security interactions: With cross-layer resilience, future adaptive systems with built-in feedback control (to optimize energy, speed, temperature, lifetime, resilience, or combinations thereof) can be vulnerable to “virus” workloads (maliciously or inadvertently launched) compromising system stability or correctness (e.g. [Bose 17]). Known gaps in device reliability may be exploited to launch malicious attacks (e.g., [Mutlu 17]). Possible (side-/covert-channel) attacks that exploit resilience techniques (or bugs introduced by their incorrect implementations) must also be explored.

ACKNOWLEDGMENT

This research was supported in part by Defense Advanced Research Projects Agency (IBM, Stanford, U. Virginia). The views, opinions and/or findings expressed are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government. Other sponsors include DFG SPP1500 (TU Munich), DTRA (Stanford), NSF (Stanford), and SRC (Stanford, UIUC).

REFERENCES

- [Agarwal 07] M. Agarwal, *et al.*, “Circuit failure prediction and its application to transistor aging,” *ITS*, 2007.
- [Aly 15] M. Aly, *et al.*, “Energy-efficient abundant-data computing: the N3XT 1,000x,” *IEEE Computer*, 2015.
- [Aly 19] M. Aly, *et al.*, “The N3XT approach to energy-efficient abundant-data computing,” *Proceedings of the IEEE*, 2019.
- [Bertran 12] R. Bertran *et al.*, “Systematic energy characterization of CMP/SMT processor systems via automated micro-benchmarks,” *MICRO*, 2012.
- [Borkar 05] S. Borkar, “Designing reliable systems from unreliable components: the challenges of transistor variability and degradation,” *IEEE Micro*, 2005.
- [Bose 17] P. Bose and A. Buyuktosunoglu, “Resilient and energy-secure power management,” *ISLPED*, 2017.
- [Capello 14] F. Capello, *et al.*, “Toward exascale resilience: 2014 update,” *Supercomputing Frontiers and Innovations*, 2014.
- [Carter 10] N.P. Carter, H. Naeimi, and D.S. Gardner, “Design techniques for cross-layer resilience,” *DATE*, 2010.
- [Baba 09] H. Baba and S. Mitra, “Testing for transistor aging,” *ITS*, 2009.
- [Chandramoorthy 19] N. Chandramoorthy, *et al.*, “Resilient low voltage accelerators for high energy efficiency,” *HPCA*, 2019.
- [Cheng 16] E. Cheng, *et al.*, “CLEAR: cross-layer exploration for architecting resilience - combining hardware and software techniques to tolerate soft errors in processor cores,” *DAC*, 2016.
- [Cheng 17] E. Cheng, *et al.*, “Cross-layer resilience in low-voltage digital systems: key insights,” *ICCD*, 2017.
- [Cheng 18a] E. Cheng, *Cross-layer resilience to tolerate hardware errors in digital systems*, Ph.D. dissertation, Stanford University, 2018.
- [Cheng 18b] E. Cheng, *et al.*, “Tolerating soft errors in processor cores using CLEAR (cross-layer exploration for architecting resilience),” *IEEE Trans. CAD*, 2018.
- [Cho 12] H. Cho, L. Leem, and S. Mitra, “ERSA: error resilient system architecture for probabilistic applications,” *IEEE Trans. CAD*, 2012.
- [Cho 13] H. Cho, *et al.*, “Quantitative evaluation of soft error injection techniques for robust system design,” *DAC*, 2013.
- [Cho 17] H. Cho, *et al.*, “System-level effects of soft errors in on-chip components,” *IEEE Trans. CAD*, 2017.
- [DeHon 10] A. DeHon, H.M. Quinn, N.P. Carter, “Vision for cross-layer optimization to address the dual challenges of energy and reliability,” *DATE*, 2010.
- [DIN 16] Safety of machinery - Functional safety of safety-related electrical, electronic and programmable electronic control systems EN 62061, 2016.
- [Grossi 19] A. Grossi, *et al.*, “Resistive RRAM endurance: array-level characterization and correction techniques targeting deep learning applications,” *IEEE Trans. Electron Devices*, 2019.
- [Gupta 14] M.S. Gupta, J.A. Rivers, L. Wang, and P. Bose, “Cross-layer system resilience at affordable power,” *IRPS*, 2014.
- [Henkel 14] J. Henkel, *et al.*, “Multi-layer dependability: from microarchitecture to application level,” *DAC*, 2014.
- [Huang 84] K.-H. Huang and J.A. Abraham, “Algorithm based fault tolerance for matrix operations,” *IEEE Trans. Computers*, 1984.
- [Inoue 08] H. Inoue, Y. Li, and S. Mitra, “VAST: virtualization-assisted concurrent autonomous self-test,” *ITC*, 2008.
- [ISO 11] ISO Road vehicles -- Functional safety. (ISO 26262), *ISO, Geneva, Switzerland* 2011.
- [Kim 10] Y.M. Kim, *et al.*, “Low-cost gate oxide early-life failure detection in robust systems,” *Symp. VLSI Circuits*, 2010.
- [Kim 13] Y.M. Kim, *et al.*, “Detection of early-life failures in high-k metal-gate transistors and ultra low-k inter-metal dielectrics,” *CICC*, 2013.
- [Lee 10] H.-H.K. Lee, *et al.*, “LEAP: layout design through error-aware transistor positioning for soft-error resilient sequential cell design,” *IRPS*, 2010.
- [Lefurgy 11] Lefurgy, C., *et al.*, “Active management of timing guardband to save energy in POWER7,” *MICRO*, 2011.
- [Le 19] B. Le, *et al.*, “Resistive RAM with multiple bits per cell: array-level demonstration of 3 bits per cell,” *IEEE Trans. Electron Devices*, 2019.
- [Li 08] Y. Li, S. Makar, and S. Mitra, “CASP: concurrent autonomous chip self-test using stored test patterns,” *DATE*, 2008.
- [Li 09] Y. Li, O. Mutlu, and S. Mitra, “Operating system scheduling for efficient online self-test in robust systems,” *ICCAD*, 2009.
- [Li 10] Y. Li, *et al.*, “Concurrent autonomous self-test for on-chip components in system-on-chips,” *ITS*, 2010.
- [Li 13a] Y. Li, *Online self-test, diagnostics, and self-repair for robust system design*, Ph.D. dissertation, Stanford University, 2013.
- [Li 13b] Y. Li, *et al.*, “Self-repair of on-chip components in robust system-on-chips: an openSPARC T2 case study,” *ITC*, 2013.
- [Lilja 13] K. Lilja, *et al.*, “Single-event performance and layout optimization of flip-flops in a 28-nm bulk technology,” *IEEE Trans. Nuclear Sc.*, 2013.
- [Lovellette 02] M.N. Lovellette, *et al.*, “Strategies for fault-tolerant, space-based computing: lessons learned from the ARGOS testbed,” *Aerospace Conf.*, 2002.
- [Lu 82] D.J. Lu, “Watchdog processors and structural integrity checking,” *IEEE Trans. Computers*, 1982.
- [Mahmood 88] A. Mahmood and E.J. McCluskey, “Concurrent error detection using watchdog processors – a survey,” *IEEE Trans. Computers*, 1988.
- [McCluskey 00] E.J. McCluskey and C.-W. Tseng, “Stuck-fault tests vs. actual defects,” *ITC*, 2000.
- [Mintarno 11] E. Mintarno, *et al.*, “Self-tuning for maximized lifetime energy-efficiency in the presence of circuit aging,” *IEEE Trans. CAD*, 2011.
- [Mirkhani 14] S. Mirkhani, *et al.*, “Rethinking error injection for effective resilience,” *ASP-DAC*, 2014.
- [Mirkhani 15] S. Mirkhani, B. Samynathan, and J. Abraham, “In-depth soft error vulnerability analysis using synthetic benchmarks,” *ITS*, 2015.
- [Mitra 14] S. Mitra, *et al.*, “The resilience wall: Cross-layer solution strategies,” *VLSI-TS4*, 2014.
- [Mueller-Gritschneider 18a] D. Mueller-Gritschneider, *et al.*, “ETISS-ML: a multi-level instruction set simulator with rtl-level fault injection support for the evaluation of cross-layer resiliency techniques,” *DATE*, 2018.
- [Mueller-Gritschneider 18b] D. Mueller-Gritschneider, *et al.*, “Performance and accuracy in soft-error resilience evaluation using the multi-level processor simulator ETISS-ML,” *ICCAD*, 2018.
- [Mutlu 17] O. Mutlu, “The RowHammer Problem and Other Issues We May Face as Memory Becomes Denser,” *DATE* 2017.
- [OpenSPARC] Oracle, “Opensparc overview,” <http://www.oracle.com/technetwork/systems/opensparc/index.html>.
- [Pedram 12] M. Pedram, D. Brooks, and T. Pinkston, *Report for the NSF workshop on cross-layer power optimization and management*, NSF, 2012.
- [Sauer 13] M. Sauer, *et al.*, “Early-life failure detection using SAT-based ATPG,” *ITC*, 2013.
- [Sinharoy 11] B. Sinharoy, *et al.*, “IBM POWER7 multicore server processor,” *IBM Journal R&D*, 2011.
- [Tschanz 09] J. Tschanz, *et al.*, “Tunable replica circuits and adaptive voltage-frequency techniques for dynamic voltage, temperature, and aging variation tolerance,” *Symp. VLSI Circuits*, 2009.
- [Vega 18] A. Vega, A. Buyuktosunoglu, and P. Bose, “Towards ‘smarter’ vehicles through cloud-backed swarm cognition,” *Intelligent Vehicles Symp.*, 2018.
- [Whatmough 18] P. Whatmough, S.K. Lee, D. Brooks, and G.-Y. Wei, “DNN engine: a 28-nm timing-error tolerant sparse deep neural network processor for iot applications,” *IEEE JSSC*, 2018.
- [Wu 18] T. Wu, *et al.*, “Hyperdimensional computing exploiting carbon nanotube FETs, resistive RAM, and their monolithic 3D integration,” *IEEE JSSC*, 2018.
- [Wu 19] T. Wu, *et al.*, “A 43pJ/cycle non-volatile microcontroller with 4.7μs shutdown/wake-up integrating 2.3 bits-per-cell resistive RAM and resilience techniques,” *ISSCC*, 2019.