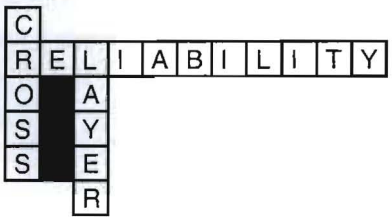
LA-UR- 10-08387

Approved for public release; distribution is unlimited.

Title:	Final Report for CCC Cross-Layer Reliability Visioning Study
Author(s):	Heather Quinn (LANL), Andre DeHon (UPenn), Nick Carter (Intel)
Intended for:	www.relxlayer.org

Form 836 (7/06)



Final Report for CCC Cross-Layer Reliability Visioning Study

André DeHon, Nick Carter and Heather Quinn, *Editors*

December 20, 2010

Contributors: Sarita Adve, Marcos Aguilera, Carl Anderson, Paul Armijo, Todd Austin, Sankar Basu, Lori Bechtold, Shawn Blanton, Shekhar Borkar, Younes Boulghassoul, Keith Bowman, Greg Bronevetsky, James Browne, Nicholas Carter, Vikas Chandra, Tim Cheng, Pierre Chor-Fung Chia, Lewis Cohn, John Daly, Chitaranjan Das, J.L. de Jong, Nathan DeBardeleben, Erik deBenedictis, André DeHon, Eliezer Dekel, Bill Eklow, Glenn A Forman, Armando Fox, Tim Gallagher, Donald S. Gardner, Kinshuk Govil, John Gustafson, Eric Hannah, William Harrod, William Heidergott, John Hiller, Andrew Huang, Ravi Iyer, David Kaeli, Zbigniew Kalbarczyk, Kevin Kemp, Prabhakar Kudva, Kimmo Kuusilinna, Shih-Lien Lu, James Lyke, William M. Jones Jr, Nikil Mehta, Sarah Michalak, Subhasish Mitra, Claude Moughanni, Shubu Mukherjee, Helia Naeimi, Sani Nassif, Suriyaprakash Natarajan, Eugene Normand, Kevin Nowka, Ishwar Parulkar, Karthik Pattabiraman, Mark Porter, Heather Quinn, Charles Recchia, Anthony Reipold, Pia Sanda, Sumeet Sandhu, John Savage, Bianca Schroeder, Sanjit Seshia, Allan Silburt, James Smith, Rafi Some, Daniel Sorin, Jon Stearley, Gary Swift, David Tennenhouse, Chandra Tirumurti, Steve Trimberger, Ian Troxel, David Walker, Shi-Jie Wen, Chris Wilkerson, Alan Wood, Vivian Zhu

Disclaimer

The material in this document reflects the collective views, ideas, opinions and findings of the study participants only, and not those of any of the universities, corporations, or other institutions with which they are affiliated. Furthermore, the material in this document does not reflect the official views, ideas, opinions and/or findings of National Science Foundation, the Computing Research Association, or of the United States government.

Organization and Contributions

Organizers:

- Nicholas P. Carter, Intel Corporation
- André DeHon, University of Pennsylvania
- Heather M. Quinn, Los Alamos National Laboratory

Core Working Group:

- Sarita V. Adve, University of Illinois Urbana-Champaign
- Todd Austin, University of Michigan
- Andrew "Bunnie" Huang, Chumby Industries, Inc.
- Ravishankar K. Iyer, University of Illinois Urbana-Champaign
- Subhasish Mitra, Stanford University
- Sani Nassif, IBM
- John Savage, Brown
- David Walker, Princeton University
- Gary Swift, Xilinx Corporation

Focus Group Leaders:

- Commercial and Consume Electronics – Todd Austin, University of Michigan
- Infrastructure – Zbigniew Kalbarczyk, University of Illinois Urbana-Champaign
- Life Critical – Mark Porter, Medtronics
- Aerospace – Heather M. Quinn, Los Alamos National Laboratory
- Large-Scale Systems
  - Greg Bronevetsky, Lawerence Livermore National Laboratory
  - Armando Fox, University of California at Berkeley
  - Sarah Michalak, Los Alamos National Laboratory
- Roadmap – Sani Nassif, IBM
- Metrics
  - Subhasish Mitra, Stanford University
  - Pia Sanda, IBM

Wiki Maintenance:

- Nikil Mehta, California Institute of Technology

CCC Liasons:

- David Kaeli, Northeastern University
- David Tennenhouse, New Venture Partners

Participants: For full list see Page 80.

Forward

This document reflects the thoughts of a group of researchers from universities, industry, and research laboratories on potential avenues of research to change how reliability is addressed in the world-wide computing infrastructure. This study and the material assembled is based upon work supported by the National Science Foundation under Grant 0637190 to the Computing Research Association for the Computing Community Consortium. The report itself is the direct result of meetings over 2009 with these researchers.

The goal of the study was to define a new vision for the future of reliable system design, and not to propose a potential reliability method. Further, the report itself was assembled in just a few months in 2010 from input by the participants. As such, all inconsistencies reflect either areas where there really are significant open research questions, or misunderstandings by the editors. There was, however, complete agreement about the key challenges that surfaced from the study, and the potential value that sharing reliability tasks across the entire computing stack might advance the field of reliable computing.

We are grateful to work with so many dedicated people that put in many long hours to help the study along. David Kaeli and David Tennenhouse helped sheppard us through the CCC process. Todd Austin, Zbigninew Kalbarczyk, Mark Porter, Greg Bronevetsky, Armando Fox, Sarah Michalak, Sani Nassif, and Pia Sanda helped lead the focus groups which helped us gain an understanding of where particular research communities were, updating the ITRS roadmap, and researching what metrics were needed. Nikil Mehta helped keep the wiki updated and running. We are grateful for all of your help.

We are honored to have been part of this study, and wish to thank the study members for their time, their effort, and their insight.

André DeHon, Nick Carter, and Heather Quinn

Contents

List of Figures

1 Executive Summary

The geometric rate of improvement of transistor size and integrated circuit performance known as Moore's Law has been an engine of growth for our economy, enabling new products and services, creating new value and wealth, increasing safety, and removing menial tasks from our daily lives. Affordable, highly integrated components have enabled both life-saving technologies and rich entertainment applications. Anti-lock brakes, insulin monitors, and GPS-enabled emergency response systems save lives. Cell phones, internet appliances, virtual worlds, realistic video games, and mp3 players enrich our lives and connect us together. Over the past 40 years of silicon scaling, the increasing capabilities of inexpensive computation have transformed our society through automation and ubiquitous communications.

Looking forward, **increasing unpredictability threatens our ability to continue scaling integrated circuits at Moore's Law rates.** As the transistors and wires that make up integrated circuits become smaller, they display both greater differences in behavior among devices designed to be identical and greater vulnerability to transient and permanent faults. Conventional design techniques expend energy to tolerate this unpredictability by adding safety margins to a circuit's operating voltage, clock frequency or charge stored per bit. However, the rising energy costs needed to compensate for increasing unpredictability are rapidly becoming unacceptable in today's environment where power consumption is often the limiting factor on integrated circuit performance and energy efficiency is a national concern. Reliability and energy consumption are both reaching key inflection points that, together, threaten to reduce or end the benefits of feature size reduction.

To continue beneficial scaling, we must use a *cross-layer, full-system-design approach to reliability*. Unlike current systems, which charge every device a substantial energy tax in order to guarantee correct operation in spite of rare events, such as one high-threshold transistor in a billion or one erroneous gate evaluation in an hour of computation, cross-layer reliability schemes make *reliability management a cooperative effort across the system stack*, sharing information across layers so that they only expend energy on reliability when an error actually occurs. Figure 1 illustrates an example of such a system that uses a combination of information from the application and cheap architecture-level techniques to detect errors. When an error occurs, mechanisms at higher levels in the stack correct the error, efficiently delivering correct operation to the user in spite of errors at the device or circuit levels.

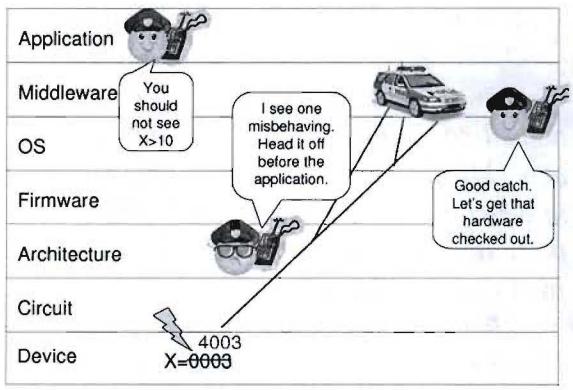


Figure 1: Cross-Layer Cooperation

In the realms of memory and communication, engineers have a long history of success in tolerating unpredictable effects such as fabrication variability, transient upsets, and lifetime wear using information sharing, limited redundancy, and cross-layer approaches that anticipate, accommodate, and suppress errors. Networks use a combination of hardware and software to guarantee end-to-end correctness. Error-detection and correction codes use additional information to correct the most common errors, single-bit transmission errors. When errors occur that cannot be corrected by these codes, the network protocol requests re-transmission of one or more packets until the correct data is received. Similarly, computer memory systems exploit a cross-layer division of labor

to achieve high performance with modest hardware. Rather than demanding that hardware alone provide the virtual memory abstraction, software page-fault and TLB-miss handlers allow a modest piece of hardware, the TLB, to handle the common-case operations on a cycle-by-cycle basis while infrequent misses are handled in system software.

Unfortunately, mitigating logic errors is not as simple or as well researched as memory or communication systems. This lack of understanding has led to very expensive solutions. For example, triple-modular redundancy masks errors by triplicating computations in either time or area. This mitigation methods imposes a 200% increase in energy consumption for *every* operation, not just the uncommon failure cases.

At a time when computation is rapidly becoming part of our critical civilian and military infrastructure and decreasing costs for computation are fueling our economy and our well being, we cannot afford increasingly unreliable electronics or a stagnation in capabilities per dollar, watt, or cubic meter. If researchers are able to develop techniques that tolerate the growing unpredictability of silicon devices, Moore's Law scaling should continue until at least 2022. During this 12-year time period, transistors, which are the building blocks of electronic devices, will scale their dimensions (*feature sizes*) from 45nm to 4.5nm. This additional scaling will be possible only if we can mitigate unpredictability and noise effects in small feature-size devices as effectively as we have been able to compensate for these effects in memory—that is, without paying an increasing energy overhead. The challenge for the near future, then, is to achieve the density and full energy benefits of ideally-scaled smaller technologies with the reliability of our larger and older technologies. Over the longer term, techniques to tolerate unpredictable behavior at low energy costs will also be necessary to make post-silicon technologies, such as quantum and molecular computation, feasible.

Several corporations shared their perspective on how reliability affects them economically:

"Microprocessor and computer system reliability is a critical issue for Intel. As we look forward, we are deeply concerned about the techniques used to tolerate errors, device variations, and silicon aging and their impact on the cost, performance, and power consumption of our products. Meeting the reliability challenges of future products will require innovative new approaches in which the entire system contributes to overall reliability, and we strongly endorse research into these architectures and methods." – Justin Rattner, Vice-President and Chief Technology Officer, Intel Corporation.

"Reliability and fault tolerance are essential to the performance of many embedded systems. For example, automotive, industrial and medical applications may involve safety-critical functionality and harsh operating environments. As the scope and complexity of these applications continue to increase, new approaches are needed to design, validate and qualify highly reliable and resilient embedded systems." – Ken Hansen, Vice President and Chief Technology Officer, Freescale Semiconductor Inc.

"Achieving high reliability in the types of high end systems that IBM makes is a very high priority for us. We invest significant design and engineering resources into the hardware, firmware and software aspects of error detection, correction, and avoidance. As the potential for these errors increases due to technology scaling, additional investments must be made, and new ideas proposed to find solutions that are cost effective, practical and achievable within current design paradigms. This is not a problem that we expect to go away anytime soon, and solving it must have a high priority." – Carl J. Anderson, IBM Fellow.

To better understand the reliability challenges facing future electronic systems, the limitations of current approaches, and the opportunities offered by cross-layer techniques that distribute the

responsibility for reliability across the entire system stack, the Computing Community Consortium funded a study into cross-layer reliability, which was carried out from March-October of 2009. All told, eighty people participated in one or more of the study group's three meetings, representing academia, industry, and government (Section A.2). During the study, we formulated constituency groups on commercial and consumer electronics, infrastructure, life-critical, space/avionics, and large-scale systems. These groups covered computing systems ranging from pacemakers and hand-held computers to satellites and supercomputers, as well as applications ranging from entertainment to protecting human life. Each of the groups had different perspectives on how scaling-induced power and reliability challenges impacted them and were willing to make different tradeoffs to address those challenges. Nonetheless, all of the industries represented are feeling pain from these reliability problems and agree that the coming challenges will not be satisfied by current solutions.

Throughout the study, participants described *situations where the* **lack of global information** *caused designers to make worst-case assumptions at individual design layers, resulting in systems that were over-designed, inefficient, and overly expensive.* Looking forward, many of the participants identified cases where current design techniques would be unable to meet the needs of future systems, making a new approach to reliable system design necessary instead of merely desirable. **Common challenges that should be addressed include:**

- *Late-Bound Information:* Critical information about how a system or component will be used and/or the characteristics of the technology that will be used to manufacture it is often unavailable until late in the design process. Inflexible system designs cannot adapt to this information as it becomes available, resulting in over-design, limited ability to use the system in different contexts, and occasional need for complete re-design late in the design process.
- *Lack of Instantaneous Operational Information:* Error rates in electronic systems vary substantially with operating conditions, such as temperature and altitude. Systems that cannot sense and react to changes in their environment must always inefficiently assume worst-case conditions.
- *Lack of Information on Application Requirements:* Different applications (mp3 player vs. power plant control) have very different reliability requirements. Hardware that does not know an application's needs must provide complete reliability even if the application can tolerate errors.
- *Lack of Information on Health of Components* and *Lack of Information on Components from Heterogeneous Suppliers:* Electronic components vary from manufacturer to manufacturer, from part to part from the same manufacturer, and from time to time on the same part. Systems that cannot analyze and adapt to these differences must assume worst-case behavior, leading to designs with poor performance and high power consumption. Software that is unaware of the state and health of the hardware it is running on cannot make intelligent decisions about its trustworthiness or appropriate use. This further suggests a need to address the *Granularity of Adaptation and Repair.*
- *Incomplete Information on Reliability and Weaknesses:* Current design methodologies make it difficult to understand the real sources of weaknesses in a design, leading to over-design of components that are not critical to reliability.
- *Analog and Passive Elements:* Analog, discrete, and passive components are critical to the data input and output paths on mixed-signal systems and can often be the weak link for data and control integrity. Systems that are unaware of the health of these devices and unable to compensate for their failure or changes will have limited reliability.

The common causes of these challenges suggest a pressing need to develop a better **scientific and engineering** understanding of *information sharing and exploitation across the multi-layer system stack that supports computations*. Specific components of this inquiry should include the following set of fundamental questions:

1. How do we design hardware and software organizations that are prepared for repair?
2. What is the right amount of error filtering at each level in the stack—from circuits through application software—and what are the best techniques for filtering?
3. How do we formulate, analyze, and manage multilevel trade-offs for fault mitigation that generalize the idea of hardware-software trade-offs, including the interfaces for cross-layer information sharing?
4. What is a theory and design pattern set for efficient, light-weight error checking that exploits the high-level properties of hardware architectures, software applications, and algorithms?
5. What is a theory and practical framework for expressing and reasoning about differential reliability, including both application needs and hardware/system organization to meet those needs?
6. How do we design scalable system solutions that can adapt to varying error rates and reliability demands?
7. How do we design components and systems that degrade gracefully and systems that are aware of their overall health?

Developing this understanding will directly impact many key **national missions**, including:

- **Supercomputing**: In order to build the ExaFLOP supercomputers that will drive science, defense, and commerce in the 2020's while staying within the power budgets of major data centers, we must increase computational efficiency more than 250-fold while simultaneously drastically reducing the fraction of time spent handling errors and extending mean-time-between-failures (MTBF) to months or years. Architecture redesign and software cooperation will be necessary to achieve these goals.
- **Satellites**: Satellite technology supports cable television, in-theater warfighters, and science. On-board processing is necessary to optimize the limited communication bandwidth to the ground, but must operate within power limits of 20–30 Watts. As radiation-hardened technologies lag commercial technologies by several generations, we must find ways to use more advanced and energy-efficient commodity technologies without sacrificing system reliability.
- **Medical**: Reliable, ultra-low-power computing systems will enable a host of breakthroughs in medical technology, including personal genomics, sensing devices that help compensate for blindness, assistive technologies for the elderly, and implantable devices that operate for years without failure or the need for recharging.
- **Commercial Industry** Our commercial and financial infrastructures all depend critically on reliable computation. The current economic recovery demands that reliable computation continue in spite of financial austerity measures.
- **Transportation**: Advanced safety features and drive-by-wire control demand the greater computational capabilities of advanced technologies, but also have even higher safety requirements to safeguard human life.
- **Security**: Cross-layer reliability techniques provide the foundation necessary to support the security needs of electronic commerce, electronic medical records, and military applications.

A **research program** in the nascent area of cross-layer reliable systems could have tremendous impact and influence over the development of this critical technology. Because of the cross-cutting

nature of this area, it will be essential for this program to enable collaboration of researchers with a wide range of expertise. The program should focus on developing example systems and the standard, open platforms that will enable the subsequent engagement of a larger community of domain experts across academia, government, and industry. It should also support and encourage the development of tools to model and characterize cross-layer reliable systems.

Government leadership is essential. The work necessary to achieve cross-layer reliable systems crosses the entire computing system ecosystem from integrated circuits to software applications. Therefore, no one vendor or research laboratory will be able to effect change by themselves. Wide-scale cooperation across specialties and organizations is necessary to revolutionize computing systems in this manner, otherwise the community will be facing yet another stop-gap revision that will only postpone these problems for a few more years! United States defense and civilian infrastructures—communication, finance, transportation, health—all depend critically on reliable operation, and the *government plays a key role in funding and providing pieces of this infrastructure, and in advocating and enforcing standards to enhance consumer safety.* Computer Engineering curricula must change to equip young engineers and programmers to design and develop reliable computing systems. Furthermore, Moore's Law scaling and the creation of value through new capabilities harnessing advanced computing have been a key **engine of economic growth** raising our standard of living in the United States. The research outlined above will provide US companies with the technologies required to sustain innovation and develop reliable products that will continue to bolster our economy and create high-value domestic jobs.

2 Vision

The electronics industry is rapidly approaching two inflection points that will force radical changes in the way integrated circuits are designed and built. The first inflection point is one of *reliability and predictability*. Transient and permanent faults, device variation, and aging will force designers to abandon current assumptions that transistors, wires, and other circuit elements will function perfectly over the entire lifetime of a system. Instead, systems will have to permit devices to fail and vary over time while delivering error-free functionality to the user.

The second inflection point is one of *energy*. Power consumption is rapidly replacing physical size as the factor that most limits the number of transistors and wires that can be integrated onto a chip. Even when power does not limit what designers can do, the energy consumed by integrated circuits is a critical issue in mobile systems, where it determines battery life, and in large-scale computing centers, where the cost of electricity is a major component of the cost of operation.

To continue improving the performance and cost of electronics at Moore's Law rates as we approach these inflection points, we must dramatically change how computing systems are designed. Currently, most computer and software systems demand perfectly-manufactured devices that do not change over their lifetimes and work in a wide range of environments. To create the illusion of perfect, unchanging devices, designers rely on "safety margins" created by operating a circuit at supply voltages and clock rates that allow correct operation even in the face of worst-case device variation and on ad-hoc mechanisms to correct the most-common errors, such as single-event upsets in memory cells.

Looking forward, this approach to reliability is not sustainable. While current approaches simplify design by making only a few levels of the system stack (typically the circuits and architecture) responsible for reliability, they achieve this simplification by imposing worst-case power and performance overheads even when a circuit is operating correctly and the devices that make up the circuit are experiencing little variation. As feature sizes decrease, increasing rates of faults, variation, and aging will make worst-case design unacceptable, and threaten to greatly decrease or even halt progress in electronic system performance.

Instead, we propose a *cross-layer* approach to reliability, in which *reliability management becomes a cooperative effort across the system stack* involving circuit design, architecture, firmware, operating systems, middleware, compilers, and application software (See Section 11). In a cross-layer reliable system, devices are expected to fail and to vary from their designed parameters, and the entire system stack works to correct errors and tolerate variations by *detecting* unexpected behavior, *diagnosing* the cause(s) of the unexpected behavior, *recovering* from errors so that the end user sees only correct operation, *reconfiguring* the system to prevent or reduce future errors, and *adapting* as the system's capabilities change over time. Distributing reliability throughout the system stack will allow designs to implement each aspect of reliability in the most-efficient way and to only spend extra energy when necessary to tolerate errors, variation, and aging, greatly increasing the energy-efficiency of electronic systems.

Promising initial research results point to the potential for cross-layer approaches, but full exploitation and wide-spread commercial adoption will demand a major paradigm shift in the way we engineer computing systems. Significant research is required to direct and enable this paradigm shift. We recommend strategic research investment in this area to catalyze cross-disciplinary teams to identify the opportunities, theoretical and engineering foundations, new interfaces, potential system models, and demonstration systems necessary to support this paradigm shift. At a time

when computation is rapidly becoming part of our critical civilian and military infrastructure and decreasing costs for computation are fueling our economy and our well being, we cannot afford increasingly unreliable electronics or a stagnation in capabilities per dollar, watt, or cubic meter.

In this section, we state our goals (Section 2.1), identify the key technological trends that are changing the integrated-circuit landscape and drive the need for new solutions (Section 2.2), and articulate why this yields a new challenge that must be addressed now (Section 2.3). We review conventional approaches to unpredictability and reliability (Section 2.4) and contrast that with the new and underexploited ideas that appear promising to address the new challenges (Section 2.5). We touch on what we might be able to accomplish with this new approach (Section 2.6), highlight the national and global impact of addressing these energy and reliability issues (Section 2.7), and articulate the need for government leadership (Section 2.8).

Following the story in this section, subsequent sections provide a more complete description of key elements. We start with an illustration of cross-layer cooperation in processor-based computing systems and System-on-a-Chip designs (Section 3). In Section 4, we summarize the results of constituency groups from five key industrial sectors (Appendix E) and identify common, underlying challenges. In Section 5, we expand on the promising new approaches and articulate key scientific research questions associated with each. Section 6 further explains how progress toward addressing these challenges impacts key national missions and society at large. We identify associated needs in education (Section 7) and critical questions associated with the research (Section 8). Section 9 describes the kinds of metrics necessary to quantify progress in this area and the need for work on metrics as part of an overall research program. Section 10 provides recommendations for organizing the research. Section 11 provides a more complete roundup of layers and highlights the large set of communities that should be engaged by this research. Appendix A summarizes how the study was organized. Appendix B provides a sample solicitation for research in this area, and Appendix C is a two-page description of the vision for a non-technical audience.

2.1 Goal

The traditional benefit of fabrication scaling has been a decrease in cost per user-visible functionality. This benefit comes from technological effects, such as a decreasing cost per gate and decreasing energy per gate evaluation. Unfortunately, continued scaling means we will see increasingly unpredictable devices that exhibit high variation, high rates of transient errors, and significant in-system aging and failure. If handling these increasingly unpredictable devices with traditional solutions, such as margining, means an increase in energy per gate, the new, scaled technology offers no advantage over the previous technology, as shown in the "Unmitigated" curve in Figure 2. Similarly, if mitigating reliability problems means triplicating logic and voting, the scaled technology might not offer a reduction in either of energy or area. The net result of mitigating the reliability problem using traditional approaches will be an increase in area and energy per gate. Both effects suggest it will not be economically beneficial to use the scaled technology. Consequently, we must find more economical ways to enhance system reliability above the device level to continue to exploit the benefits of further feature-size scaling.

Our goal is to facilitate the successful navigation of the energy and reliability inflection points. Specifically, this means finding solutions that maintain or improve system safety while allowing continued scaling benefits. To continue scaling, we must continue to deliver increased operations per unit time while working within a fixed power-density budget. To achieve this end, we accept

that raw device reliability and consistency will decrease and look for ways to build reliable and predictable systems from unreliable and unpredictable devices. Modern energy and power challenges demand that the mitigation techniques used to compensate for unpredictable devices be energy efficient. That is, they must require only a small energy investment and lead to net energy reductions relative to unscaled solutions, as shown in the "Goal" curve in Figure 2.

2.2 Trends

We now review a number of important trends that are changing the landscape in integrated circuit design and use. These trends set up the challenge we are addressing.

Power Density Limits Flat power density budgets, such as 100W/cm^2 for forced-air cooling or 1–10W/cm^2 for ambient cooling, coupled with increasing transistor count and slowly reducing capacitance, demand that supply voltages scale down with feature sizes. However, since the transistor sub-threshold slope does not scale and we need to maintain high I_{on}/I_{off} ratios, we cannot scale voltage down aggressively enough to meet the power density limit if all devices switch. As shown in Figure 3, as feature size decreases, the amount of power needed to activate the entire chip increases exponentially. Limited voltage scaling leads to the current inflection where the power density budget prevents us from activating all the devices we can potentially manufacture on a circuit. Nonetheless, *absent reliability concerns,* it remains possible to reduce the absolute energy required per switched device.

Increasing Variation→Increasing Margins Decreasing feature size leads to increasing variation, as noted in the ITRS [6] and shown in Figure 4. This increase in variation will have two negative effects. First, as shown in Figure 5 and further detailed in Appendix D, as feature sizes decrease, increasing rates of variation lead to decreases in the number of standard deviations of device variation a circuit can tolerate before it fails completely. Even for fixed-size circuits, the increasing variation rate will lead to increased defect rates in future fabrication processes. Second, conventional margining techniques set operating voltages to guarantee correct operation across the expected range of device characteristics found in a chip, such as three standard deviations from the mean ($\pm 3\sigma$). When the standard deviation becomes a significant fraction of nominal voltage (*e.g.* $\sigma_{V_t}/V_t \rightarrow 27\%$ before F=22nm [6]), an increasing percentage of the voltage swing must be dedicated to margining for worst-case devices. This effect further limits our ability to reduce supply voltages if the industry is to maintain current fabrication yield rates. In [26] shows an example

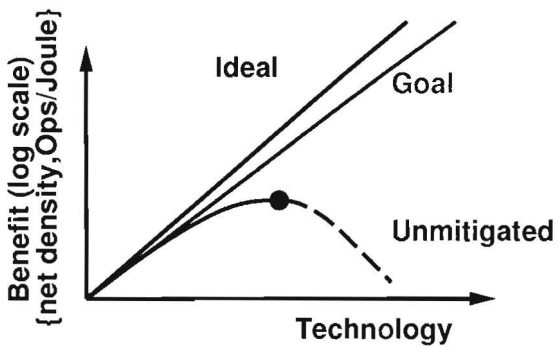


Figure 2: Scaling Scenarios

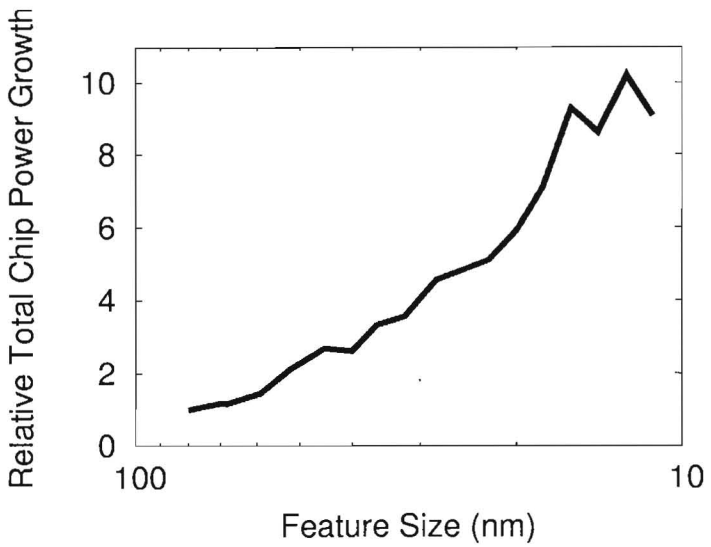


Figure 3: Projected chip power growth based on ITRS capacity, voltage, and capacitance, scaling shown relative to 90nm. Since we are already power limited, this is the energy reduction gap needed to be able to utilize the chip's potential capacity.

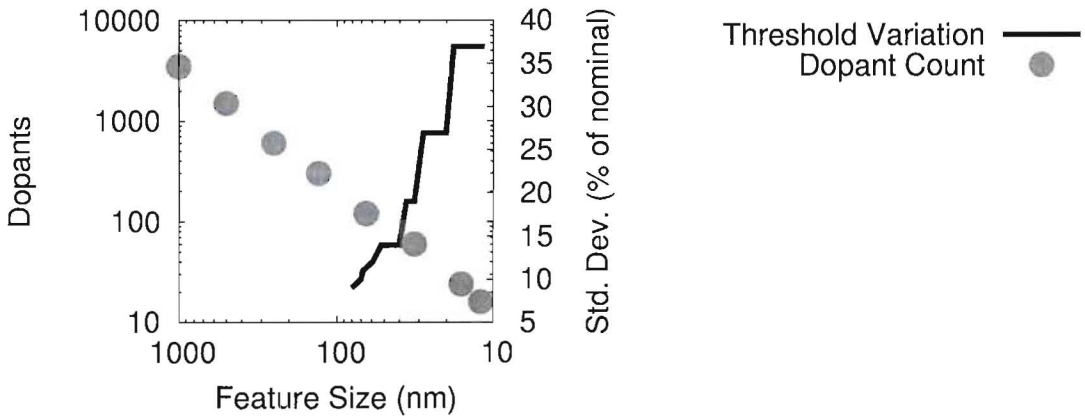


Figure 4: Shrinking feature sizes mean shrinking dopant counts which lead to higher variation in dopants, increasing random dopant variation. Feature shrinking is one of the factors increasing threshold (V_t) variability (ITRS composite V_t variation prediction shown).

where the minimum energy per operation considering the expected variation actually increases as we scale from the 45nm to the 32nm node. This example is a concrete case demonstrating the end of scaling benefits shown in the unmitigated case of Figure 2.

Increasing Device Count per Chip As fabrication processes improve, we continue to increase the number of transistors per integrated circuit, increasing the number of transistors that each chip statistically samples from the device parameter distribution. To achieve comparable chip-level yields via margining, engineers are forced to accept a larger spread of device characteristics. That is, if we needed $\pm 3\sigma$ margins to get adequate yield at smaller transistors counts, we might be forced to now tolerate $\pm 4\sigma$ margins as shown in Figure 6.

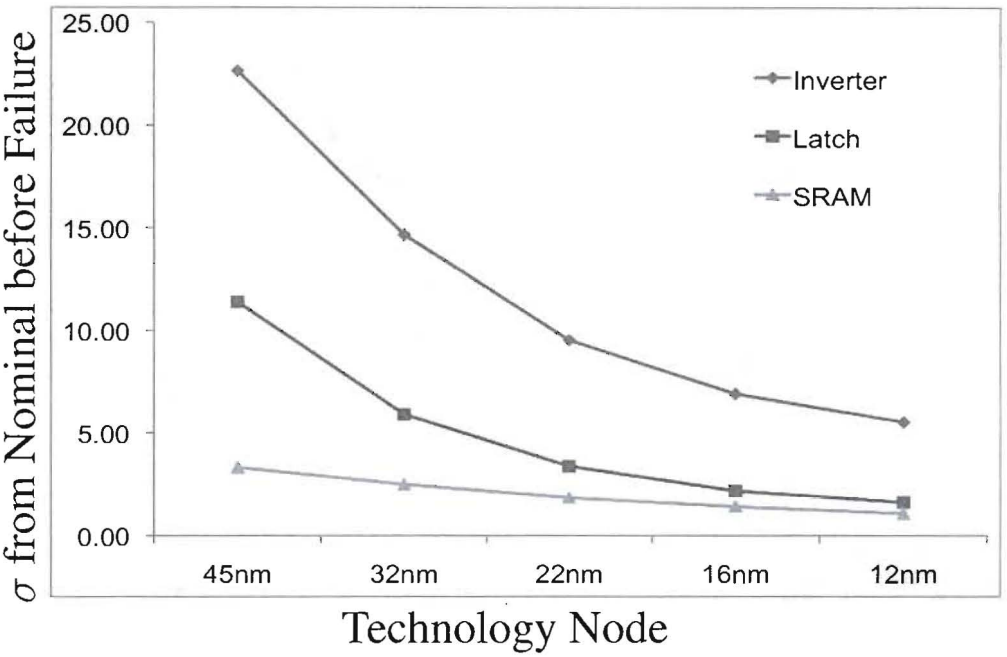


Figure 5: Failure rate **per circuit element** due to variation effects

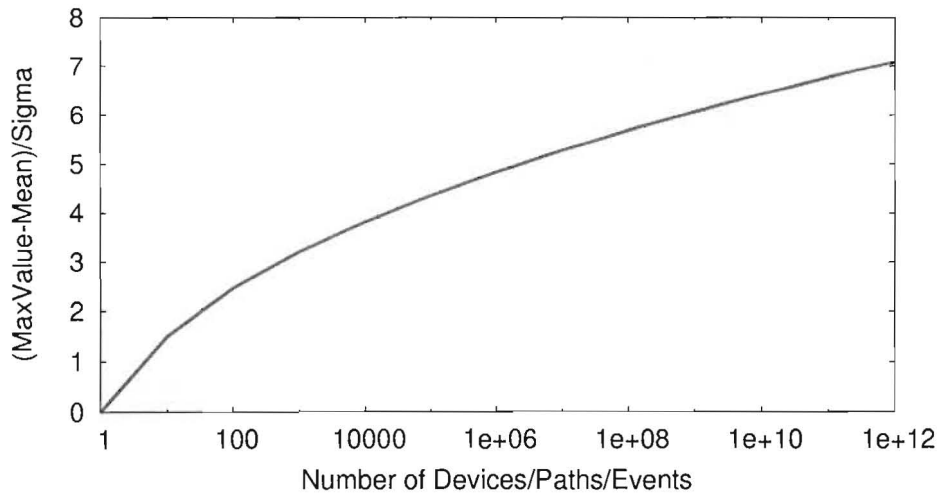


Figure 6: Increasing transistor counts means chips must tolerate variation further out in the tails of the transistor parameter distribution

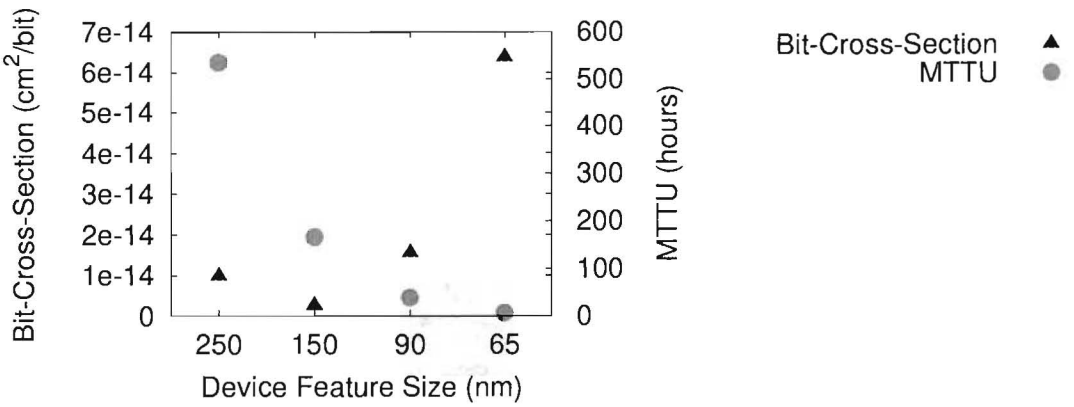


Figure 7: Mean-Time-To-Upset (MTTU) decreases as feature sizes shrink [shown at a flux of 23,082 neutrons-cm^2/hour corresponding to the flux seen by an airplane flying at 60,000 feet over the North Pole where the magnetic field lines converge]; sensitivity per bit (Bit-Cross-Section) does not show direct correlation with feature size
[99, 100]

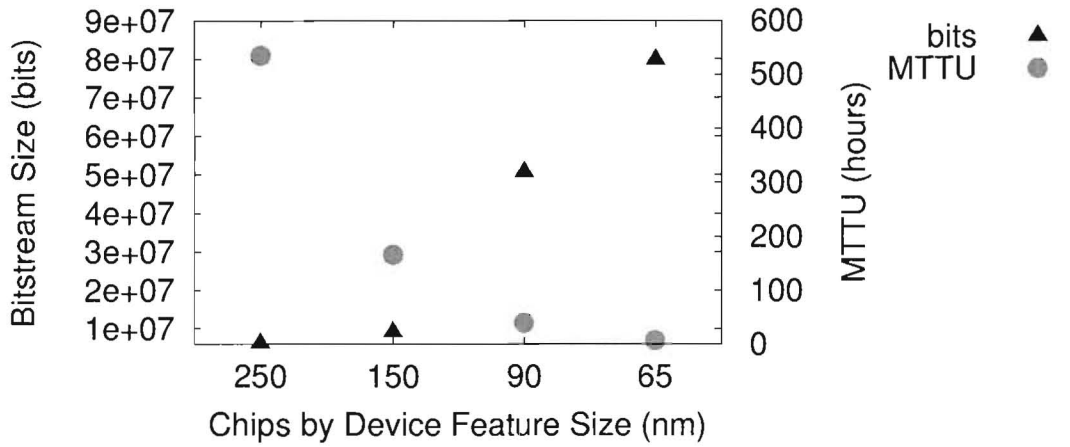


Figure 8: Chip capacity continues to increase as feature sizes shrink; capacity increase correlates well with decreasing MTTU

Increasing device count per chip also increases the opportunity for transient upsets. As an example, we will discuss the results of testing Field-Programmable Gate Arrays (FPGAs) in high radiation environments. In FPGAs, the predominant radiation problem stems from single-event upsets (SEUs) in the configuration memory (*bitstream*). Accelerated tests are used to measure the area sensitive to radiation [99, 100]. On a per-bit basis this value is called *bit-cross-section*. Bit-cross-section is a useful in comparing the sensitivity to radiation across feature sizes and manufacturing processes. It is also used to determine the per-bit error rate and the device error rate for a given chip. Figure 7 shows that while the sensitivity to radiation-induced upsets for a single memory device, as measured by the bit-cross-section area of the device, has not changed significantly over several generations, the chip-wide mean-time-to-upset (MTTU) seen during accelerated testing has decreased exponentially. As shown in Figure 8, this correlation is due to the increasing size of

the bitstream for each generation of device. Therefore, even if the individual device sensitivity to radiation decreases, the increase in the number of devices negates these effects.

Increasing Transient Upset Rate per Device Decreased feature size and voltages also mean a decrease in the amount of charge that represents a bit of state in a circuit (critical charge). Decreased critical charge increases susceptibility to thermal [69] and shot noise [68]. Numerous articles show that transient upset rates increase as voltages decrease [12, 43, 14, 30]. Higher clock rates also increase upset sensitivity [49].

Increasing Chips per System Even with geometric increases in the number of transistors per chip, the number of chips in large supercomputers [15] and data centers [18] is increasing at significant rates over time. These state-of-the-art large-scale systems will see composite increases in errors due to increased per-chip error rates and increasing system sizes, which will significantly decrease their mean-time-to-failure unless we develop better mechanisms to detect and correct errors.

Decreasing Effectiveness of Burn-In Burn-In testing, where wear is accelerated by increasing temperature and/or supply voltage, is commonly used to detect devices that would have failed early in the field. However, as feature sizes shrink, burn-in is becoming less effective [27], implying that more weak devices will escape initial test and fail during use. Similarly, increasing wearout effects, including negative-bias temperature instability [114] and hot carrier injection [106], will further increase permanent failure rates in integrated circuits or demand significantly increased safety margins with corresponding decreases in energy-efficiency [37, 125]. The cited studies predict increases in persistent error failure rates of about $2\times$ per fabrication generation, suggesting that consumer electronics at 45nm technology may already be down to 3-5 year per-chip lifetimes and could drop below one-year lifetimes in the next five years unless designs become capable of tolerating moderate numbers of device failures. This situation means we won't be buying the latest smart phone or laptop every 6–12 months because of fashion and features, but because the systems simply won't last that long.

Increasing Deployment in Critical Applications These reliability challenges come at a time when the impact of failure is increasing. Electronics are being deployed more pervasively into all aspects of our lives (*e.g.* cell phones, PDAs, business transactions), into our critical infrastructure (*e.g.* building, power grid, financial, e-commerce, communications, GPS satellites), and into life critical roles (*e.g.* automotive, aerospace, medical components). Our modern world increasingly depends on the reliable operation of a growing number of these devices, increasing both our susceptibility to and the impact of integrated circuit failure. This situation drives an increasing need for higher-reliability systems—a trend opposite of where device-level scaling is headed. The result is a widening gap between device-level reliability and system-level reliability requirements.

One area of recent concern has been the reliability of cars. The most recent car reliability standard, "Failure Mechanism Based Stress Test Qualification for Integrated Circuits (AEC-Q100-Rev-G)", states that "[Soft Error Rate (SER)] testing is needed for devices with large numbers of SRAM or DRAM cells ($\geq$ 1 Mbit). For example: Since the SER rates for a 130 nm technology are typically near 1000 FIT/Mbit, a device with only 1,000 SRAM cells will result in an SER contribution of 1 FIT." In Figure 9, the mean-time-to-upset in hours is shown for the worldwide population of cars as a function of the memory capacity in each car. These calculations take into account that approximately 250 million [89] cars are on the road every day with an average time on the road of three hours. Of the 250 million cars, we assume that the new cars are more likely to have more memory, so the MTTU is further rated to indicate that 60% of all cars on the road

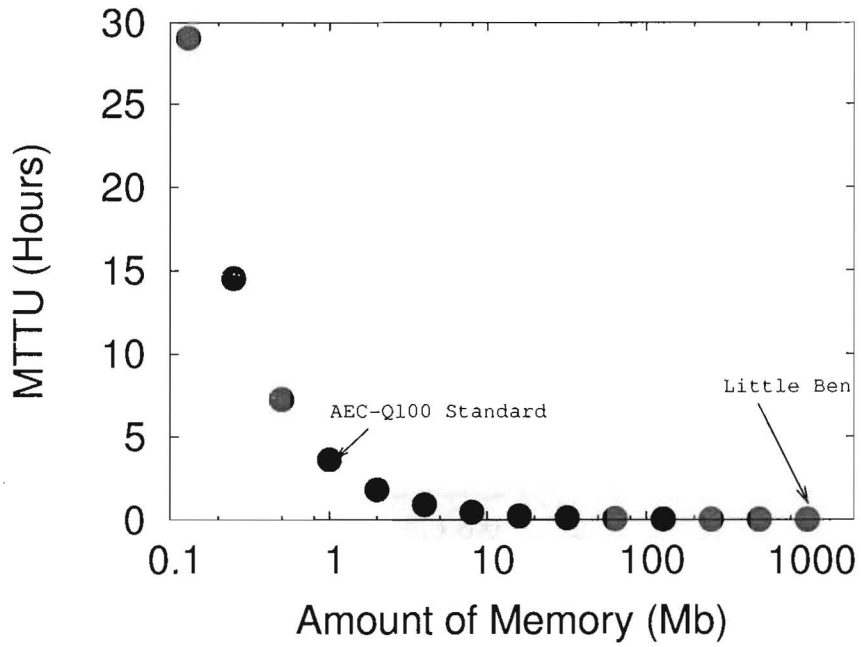


Figure 9: Worldwide mean-time-to-upset in hours for different car memory sizes

are manufactured in the last ten years. From this graph we see that if all of these cars had only 1Mb of memory, a single-event upset will occur approximately every 3.6 hours. For the computer-driven Grand and Urban Challenge [8] cars, such as Little Ben [25], that can have up to 128MB of memory, the MTTU is 12.6 seconds. These autonomous vehicles are indicative of the level of electronics we may see in cars over the next 10–20 years as automobile electronics continues to assist with more of the driving functions, such as auto parking, lane sensing, distance sensing, and cruise control.

While error-correcting codes can correct some of these errors, the increasing likelihood of multiple-bit upsets, where a single ionizing particle creates multiple memory bit flips, can make correcting the errors expensive. Depending on the layout of the memory devices, multiple-bit upsets can be as high as 90% [44] of all events. Furthermore, single-event functional interrupts that destroy entire pages of memory are now as common as single-event upsets in DRAMs [58, 28]. While there are many error-correcting codes that can correct multiple faults simultaneously, such as Reed-Solomon, often times they require a particular memory usage pattern that will make random accesses to memory very expensive.

Emerging Nanoscale and Disruptive Technologies All nanoscale technologies, including molecular electronics, quantum computing and biocomputing, that may offer benefits beyond silicon scaling exhibit similar or more extreme noise problems with defects, variations, and transient upsets. This situation arises largely because many of the small scale phenomena, such as thermal noise, quantum effects, statistical bonds and electron movement, thermodynamics of heat transfer, apply to all devices at this scale. Consequently, the payoff for finding more economical ways to combat these noise effects is not limited to extending the beneficial lifetime of silicon. Rather, it is necessary to make the exploitation of all of the contemplated post-silicon technologies viable.

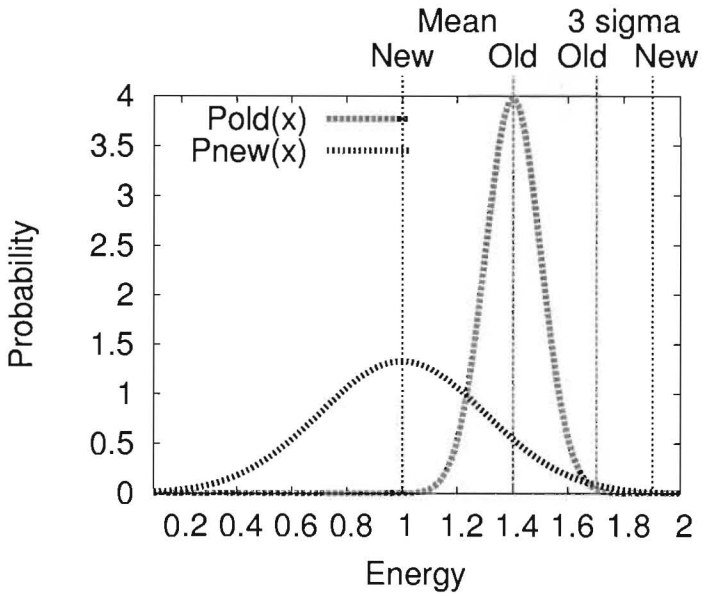


Figure 10: Mean Energy decrease, but 3σ parametric yield Energy increases

2.3 Why Now?

While reliability is not a new challenge [86] and has been an continuous concern in life-critical and harsh-environment systems, the convergence of the trends reviewed in Section 2.2 suggest that reliability will now impact **all** systems and be present on a qualitatively different scale than we have seen in the past. In particular, we are now approaching a convergence of two inflection points:

1. Energy – limits on practical power dissipation has now reached a point where energy concerns (both power density and absolute energy draw) limit the computation we can deploy on a chip. The primary driver in computational design shifts from transistor density and speed to power density and energy cost [63, 87].

2. Reliability – variation in parameters due to small scale effects coupled with larger device counts is rapidly driving the need for higher percentage margins. As shown in Figure 10, while the mean energy and delay of devices may continue to decrease with scaling, the expected worst-case devices on future chips could have higher delay and demand higher voltages for correct operation than devices constructed in current fabrication processes.

The convergence of these two inflection points presents a challenge to our status-quo approach to reliability. Our need to continue to reduce energy per device operation to increase the performance delivered per Joule or per W/cm^2 is limited by our need to provide increasing margins to deal with more variable and noisy devices, threatening an end to beneficial scaling. While it is possible to continue to produce smaller feature size components, following the traditional approach of using energy margins to hide reliability effects at the circuit level will prevent further reduction in the energy per device operation.

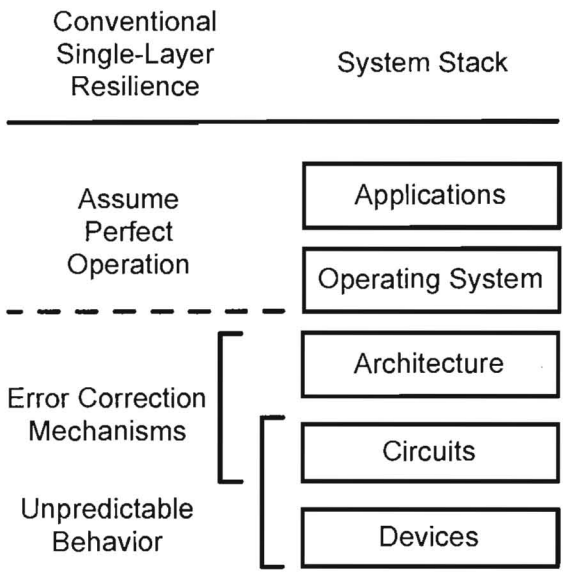


Figure 11: System Stack and Reliability

2.4 How is it done today?

Most current systems use single-layer (or few-layer) approaches to reliability that emphasize convenience over efficiency. As illustrated in Figure 11, single-layer reliable systems concentrate their reliability mechanisms in one or two layers of the system stack, allowing other layers, in particular software layers, to assume that the hardware operates reliably and predictably. The chief drawback of these approaches is that their lack of system-wide information forces them to make worst-case assumptions, leading to greater and greater inefficiency as rates of faults, variation, and aging increase with shrinking feature sizes.

The impact of these worst-case assumptions is illustrated by *margining*, the most widely-used technique for tolerating device variation and aging in current designs. Under margining, a circuit is tested to determine its maximum operating frequency at a given supply voltage. When the circuit is installed in a system, it is operated at a lower clock rate and/or higher supply voltage in order to guarantee that the circuit will still operate correctly even if aging and/or a hostile operating environment degrade its performance. In effect, the circuit is treated as if it is operating under worst-case conditions, including high temperature, worst-case device variation, or significant aging, even when conditions are the more typical controlled temperature, average-case device variation, relatively-new circuit environment, wasting either power or performance. In older fabrication processes, intra-die device variation and aging were small enough effects that margining was a reasonable choice that simplified designs at relatively-low cost. As feature sizes shrink, however, rates of intra-die variation and aging have increased to the point where margining is becoming unacceptably expensive, arguing that the industry needs new approaches to variation and aging that do not assume worst-case conditions at all times. For example, experiments with circuits that detect delay faults (errors caused when a circuit does not complete its computation by the end of a clock cycle) and correct those faults by re-executing operations have demonstrated power reductions of 30-50% by allowing systems to operate with lower clock rate and voltage margins [14, 31].

The error-handling techniques used today also emphasize convenience over efficiency, and are

predicated on the assumption that both permanent and transient errors are rare, an assumption that is becoming less and less true over time. Fabrication defects are handled through fabrication-time testing and "burning in" chips by operating them at elevated temperature and/or supply voltage for a period of time. With the exception of highly-regular structures such as RAM arrays, any device failure that is detected during test/burn-in causes the chip to be discarded. Chips that develop permanent errors during system operation must typically be discarded and replaced, although a system's error-correction mechanisms may allow it to operate with lower reliability until this is done.

Systems that require high reliability, such as mainframes and aeronautics, typically tolerate run-time hardware errors by replicating computations and comparing the results of each copy. Replication, either across multiple processors running in parallel [75, 135] or by running multiple copies of the software on the same processor [101, 109, 82, 104], allows errors to be detected and corrected very quickly, and can be implemented in ways that are invisible to application software, but has high costs. Duplicating computations to detect errors more than doubles the energy costs of the computation, since comparing the results of the two computations also costs energy, and halves the amount of computation the system can complete in a given amount of time. Systems that perform three or more copies of a computation in order to correct errors by voting have correspondingly higher overheads.

The high costs of replication have led to systems that can accept somewhat-higher rates of undetected and uncorrected errors to adopt lower-cost mechanisms, such as error-correcting codes (ECC) on RAM arrays and/or residual arithmetic for datapaths [136]. As error rates have grown, successive generations of products have incorporated more and more of these mechanisms in order to maintain constant levels of reliability. This "target the low-hanging fruit" approach to reliability can be effective when the vast majority of errors are due to a small number of causes, because a small number of mechanisms can target a large numbers of errors. As error rates and the number of physical phenomena that cause noticeable amounts of errors increase, the total cost of the mechanisms required to achieve a given system reliability increases, particularly when design and verification effort are taken into account.

2.5 What is New?—Keys to the Solution

In addition to noting that today's challenges are more acute (Section 2.3) and traditional solutions are too expensive (Section 2.4), we have identified a number of promising and under-exploited techniques that could be developed to address the new challenges. This section highlights this set of promising techniques.

A unifying theme that emerged from the study is that we are paying unreasonable overheads operating individual layers without information or assistance often available elsewhere in the abstraction stack (Section 4). *Exploiting this information is the key to enabling new solutions that achieve greater reliability with less energy overhead* (Section 5) allowing beneficial scaling to continue. These cross-layer designs cooperate to mitigate reliability problems that can no longer be addressed efficiently at the device and circuit level.

The cross-layer approach is illustrated in Figure 12. In this example the application layer determines a bound on the value x. Unlike conventional systems, the cross-layer version shares this information with lower layers. The architecture layer can use this bound to detect misbehaving hardware. However, the architecture layer does not need to capture and correct the problem. When

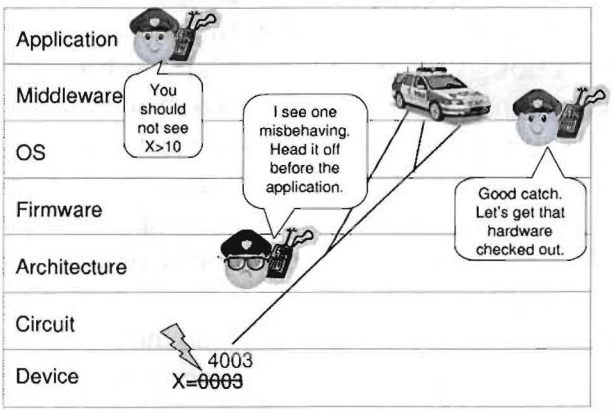


Figure 12: Cartoon Illustration of Cross-Layer Cooperation

it does notice misbehavior, it can signal the the middleware and OS layers to contain and correct the error before it becomes visible to the application. We are beginning to see scattered solutions with this flavor. Nonetheless, this approach demands a wholesale paradigm shift in the way we design and engineer computer systems.

Multi-level solutions are not without their precedent in today's computer systems. They are regularly employed to protect storage and communications. While multi-level solutions have been useful in protecting bulk storage, such as DRAMs and persistent storage, similar solutions for computation currently do not exist. In part, the heterogeneous design of computing systems and their ability to transform data makes posing simple solutions that do not rely on brute-force replication difficult. Nonetheless, there are hints of abundant opportunities for economical cross-layer protection of computations.

Hardware and software organizations must be designed for repair. Both RAM and hard disks expect errors and employ microarchitectures and abstractions that allow repair. While RAM repair, such as row and column sparing, occurs below the architectural level and is invisible to the software, bad disk sectors in hard drives are visible to the operating system. In a similar manner, our computational organizations must be prepared for errors, and must be able to dynamically reconfigure themselves to exploit late-bound information about how variation and errors are affecting the system over time [42, 52]. Mitigation of errors will likely require cooperation across the microarchitecture, architecture, and operating system.

Errors must be filtered at multiple levels—from circuits through application software. To use small devices, memory systems allow individual memory bits to fail. The microarchitecture assists by correcting errors during memory access. The operating system collaborates by scrubbing memory. To use small devices and low energy for computation, we must similarly expect occasional errors in the computation. These errors will need to be caught and corrected at higher levels in the system stack.

Multilevel trade-offs provide efficient solutions, generalizing the idea of hardware-software trade-offs. With errors slipping through devices, higher levels in the stack must be prepared to detect and correct them. Similar to the way we distribute the function of virtual address translation across hardware (*e.g.* translation lookaside buffer) and software (*e.g.* miss handling and replacement), efficient solutions will carefully divide functionality between the microarchitecture and sys-

tem software. This solution avoids paying a large energy costs for uncommon events. Suitable architectural interfaces will be required and will benefit from compiler and application support.

Light-weight error checking that exploits the high-level properties of hardware architectures, software applications and algorithms improves solution efficiency. Information theory tells us how to provide shared redundancy across large blocks of data to avoid brute-force replication. Efficient computational solutions will similarly avoid brute-force replication. For example, invariants and end-to-end consistency checks on computations may allow lightweight error detection. Characterization of the origin and reproducibility of data may allow more efficient state protection and checkpointing [50]. This further allows the hardware to safely operate on the edge of failure, using information to detect and recover when the system goes over the edge, avoiding the need to spend energy in margins to guarantee the edge is never encountered [14].

Differential reliability, including both application needs and hardware/system organization, enables more efficient solutions. DRAMs with Error-Correcting Codes (ECC) and row sparing carefully exploit the fact that the ECC allows the core of the memory to be less reliable than the periphery. This solution also exploits the ability to fabricate devices with different feature sizes to assure stronger reliability. Computations can similarly employ a mix of larger, more-reliable devices and smaller, less-reliable devices. Similarly, we can use higher voltages and currents to make some circuits more reliable than others. Differential reliability can also be realized logically, replicating or protecting one portion of the computation more heavily than others. Thereby, computations that have efficient checks or are less sensitive to errors can be run on smaller, lower-energy devices or circuits. For example, since datapath errors in a video processor only affect a few pixels while a single control error can affect a large number of pixels, it is advantageous to protect the control circuits and data more heavily than the datapath circuits and data. In this manner, high-level information about application invariants or requirements drive microarchitectural decisions around the deployment of circuits and devices with different characteristics.

As a multi-level cache memory system attempts to provide the density of a large memory with the speed of small memory:

- A traditional, ECC-protected memory provides the reliability of large feature sizes with the density of small memory cells.
- Multi-level computational designs can provide the reliability of large-feature and large-energy devices with the density and energy consumption of small-feature, low-energy devices.

Scalable systems solutions allow adaptation to error rates and reliability demands. Scalability to different error rates and different levels of protection is not present in traditional DRAM memory systems. Nonetheless, information theory does tell us how to develop codes of different rates to handle different needs, and it is easy to see how to add adaptability for memory systems. With growing error rates, error rates that vary with environment, and applications with differing needs for protection, we need the engineering understanding of how to best provide that protection across the design space as well as architectures and components that can be tuned in-system to varying environmental conditions. Device wear suggests error rates will change over time in a single component, further driving the need for in-system adaptation.

Components and systems should degrade gracefully and the system should be aware of its overall health. The system should not move from a state of correct operation to one of failure without noticing early-warning signs. It should be able to assess its readiness before performing tasks and self-report when it cannot meet the requested level of reliability.

2.6 What Can We Accomplish?

As discussed in Section 2.2, the costs of tolerating the levels of faults, variation, and aging expected in future fabrication processes threaten to erase many of the benefits of scaling to smaller feature sizes. If nothing is done, the electronics industry is facing the potential end of Moore's Law and the geometric rates of performance, energy, and cost improvements it has enjoyed for the last several decades. Cross-layer approaches to reliability offer the potential to tolerate these negative consequences of feature-size scaling with much lower overheads than current techniques, restoring the improvements in performance and energy consumption gained by reducing device feature sizes. This approach will allow the electronics industry to continue to progress at Moore's Law rates for the foreseeable future, which will in turn allow us to develop new classes of electronic systems that will save lives, protect our borders, and drive science and the US economy for decades to come. (Section 6 describes these new systems in more detail.)

To illustrate both the danger of continuing with current reliability techniques and the potential of cross-layer reliability, consider the impact of threshold voltage (V_{th}) variation on power consumption and performance. As discussed in more detail in Section D, V_{th} variation is just one of many types of device variation seen in electronics, although it is currently the one that has the greatest effect. The primary impact of threshold voltage variation on circuits is a decrease in performance (increase in delay) when high deviations make it more difficult to turn transistors on. Threshold voltage variation can also cause increases in leakage current by making it harder to turn transistors off.

Current systems tolerate threshold voltage variation through safety margins, either by increasing the supply voltage (voltage margining) so that the performance of the circuit with variation matches its nominal performance or by decreasing clock rates (clock rate margining) to allow the circuit to operate correctly in the presence of threshold voltage variation (or some combination of the two techniques). Voltage margining maintains high performance at the cost of increased power consumption, while clock rate margining decreases the performance of the circuit.

Figures 13 and 14 show how voltage margining to tolerate V_{th} variation affects the power consumed by logic circuits in a variety of fabrication technologies. Figure 13 shows the switching (active) energy consumed by a fanout-of-four (FO4) inverter at the nominal parameters of each fabrication process and when voltage margining is used to tolerate one, two, and three standard deviations (sigma) of threshold voltage variation, while Figure 14 shows the ratio of the power when tolerating each level of variation to the power at the nominal parameters for the process. Parameters for each process and variation estimates were taken from the International Technology Roadmap for Semiconductors [7].

A fanout-of-four inverter is a commonly-used benchmark circuit that consists of one inverter whose output drives four identical devices. In the graphs shown here, the N device of each inverter is sized eight times the minimum width allowed in the process, and the P device is sized to make the inverter's rising and falling delays identical. This is a typical size for devices found in production circuits, and one that avoids overstating the impact of variation on performance and power.

At the nominal parameters for each process, the switching energy of an FO4 inverter decreases from 4.55 femtojoules to 0.72 femtojoules as the fabrication process scales from 45nm to 16nm, a factor of 6.25. However, when the supply voltage of the circuit is increased to tolerate three standard deviations of V_{th} variation, as might be done in a current integrated circuit [7], the power only decreases from 5.92 femtojoules to 1.41 femtojoules as the fabrication technology scales, an improvement of only 4.20×. Using voltage margins to tolerate three standard deviations of V_{th}

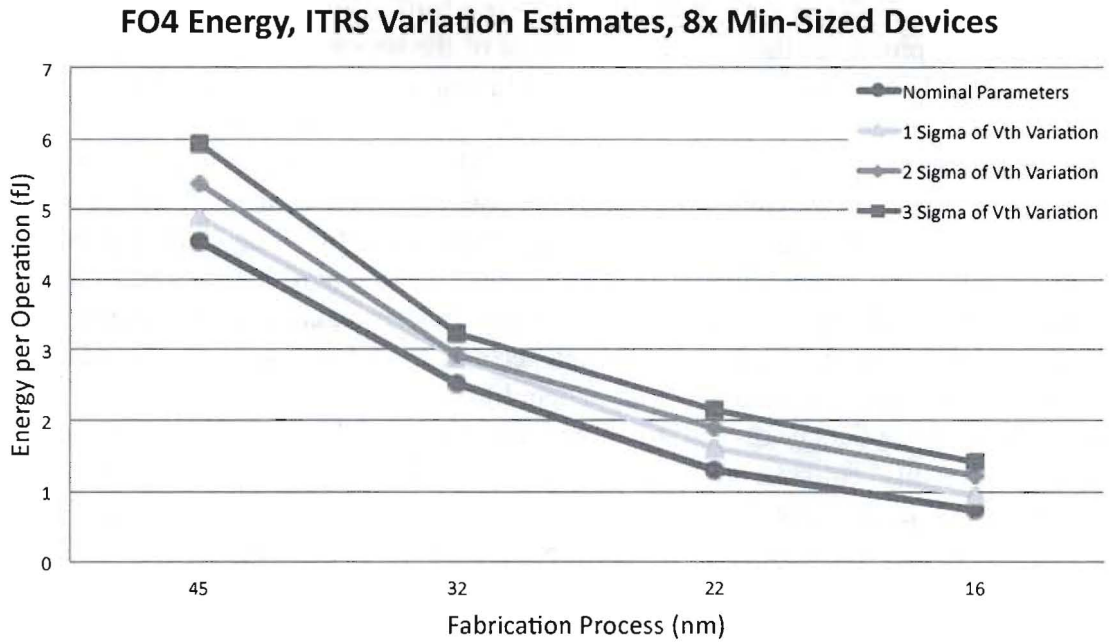


Figure 13: Impact of V_{th} Variation on Logic Energy

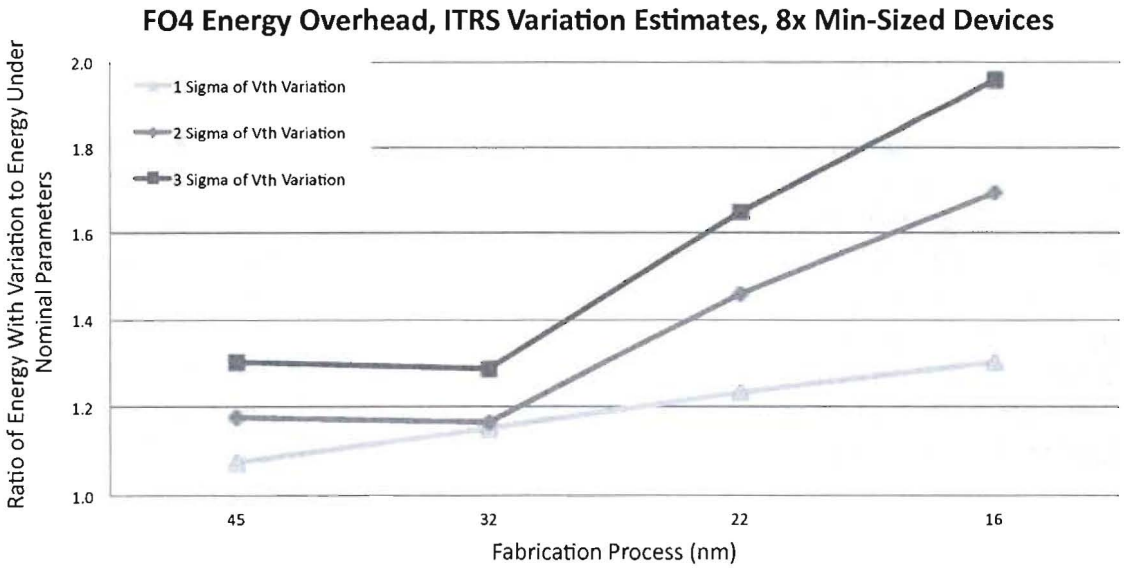


Figure 14: Ratio of Logic Energy When Margined to Tolerate V_{th} Variation to Logic Energy at Nominal V_{th}

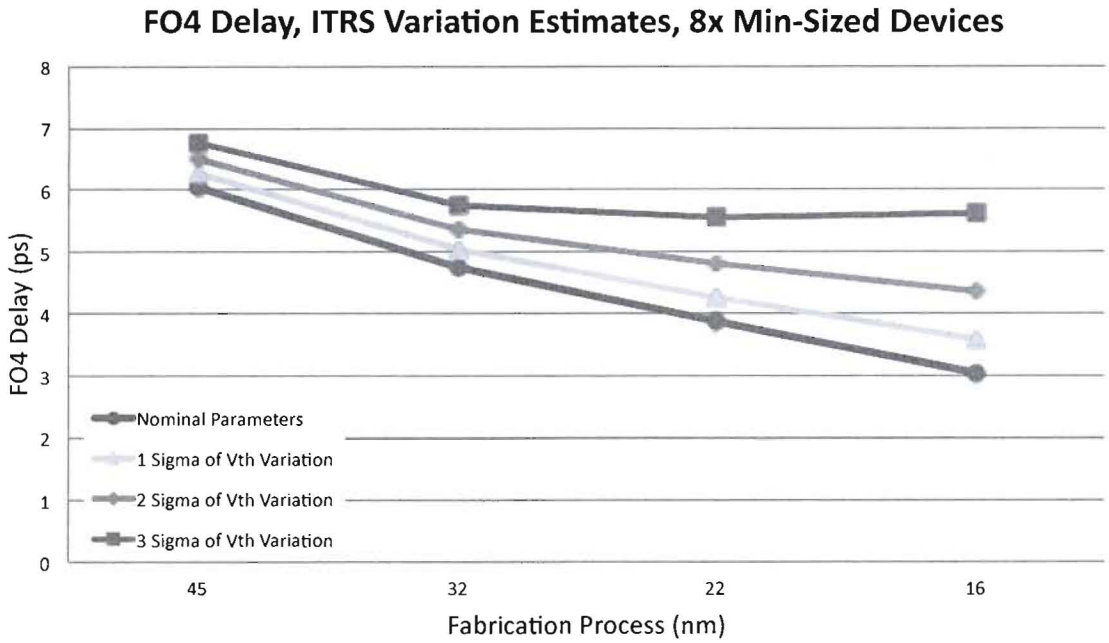


Figure 15: Impact of V_{th} Variation on FO4 Delay

variation reduces the power consumption benefits of fabrication process scaling by 33% over four technology generations. In practice, the costs of voltage margining would be expected to be even higher, because the increase in the number of devices in a circuit as fabrication processes scale would lead to an increase in the number of standard deviations of variation that chips have to tolerate through margining (Section 2.2).

Figure 14 shows that the power cost of tolerating three standard deviations of threshold voltage variation increases from 30.1% of the circuit's nominal power consumption in the 45nm process to 95.8% in the 16nm process, and also shows how cross-layer reliability can help to keep this overhead under control. Current circuits must voltage margin for the worst case that they expect to encounter because they have little or no ability to tolerate the delay faults that occur when extreme variation causes a circuit not to meet its timing requirements. A cross-layer approach might employ RAZOR latches [14] or application self-checking to correct infrequent delay faults, allowing it to operate with only one standard deviation of voltage margins. Decreasing the size of the margins would reduce the costs of voltage margining in the 16nm fabrication process to 30.2%, essentially maintaining current levels of overhead as fabrication technologies scale and allowing integrated circuits to extract almost all of the potential benefits of fabrication process scaling.

If clock rate margining is used to tolerate V_{th} variation, the costs are even more severe. As shown in Figure 15, running a circuit with three standard deviations of V_{th} variation at the nominal supply voltage for each process leads to a net *increase* in circuit delay as we move from the 22nm process to the 16nm process instead of the expected decrease. In this scenario, increasing variation negates all of the performance benefits of fabrication scaling. However, if we are able to transition from needing three standard deviations of clock rate margining in the 45nm process to only one in the 16nm process by employing other techniques to tolerate delay faults, the performance cost of clock rate margining only increases from 11.9% in the 45nm process to 18.0% in the 16nm process.

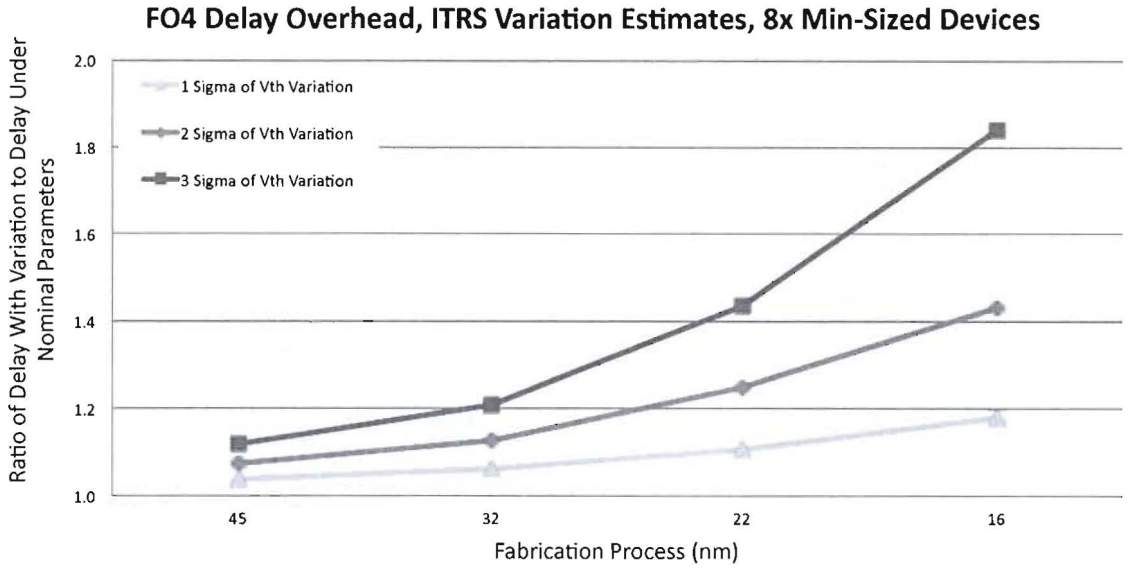


Figure 16: Ratio of FO4 Delay When Margined to Tolerate V_{th} Variation to FO4 Delay at Nominal V_{th}

Cross-layer reliability is not able to do as much to reduce the costs of clock rate margining as it was able to reduce the costs of voltage margining, but it still allows a circuit that uses clock rate margining to extract most of the potential benefits from feature-size scaling.

While this discussion has focused on threshold voltage variation, similar arguments apply to tolerating supply voltage variation [102], device width variation, and the effects that cause integrated circuits to decrease in performance as they age. The costs of tolerating the worst-case manifestations of all of these effects through margining and other current reliability techniques will rapidly become prohibitive. However, reliabilty schemes that employ multiple techniques across different levels of the system stack will be able to maintain system reliability at acceptable costs by margining for common-case behavior and efficiently correcting the errors that occur because of worst-case combinations of parameters.

2.7 Why Do This?

There are a number of reasons to fix these problems right now. As computer automation becomes a pervasive stratum for our world-wide society, the reliability of our systems will only need to increase. Currently, Moore's Law feature size scaling is often at odds with reliability, as highly scaled systems tend to be less reliable than their older counterparts. Reliability failures also make it difficult to continue to scale systems to larger sizes, as larger, more diverse systems will be increasingly susceptible to lifetime and transient failures. Reliability failures could not only affect the security of our financial systems and utility infrastructures, but could lead to human fatalities. Below, we have highlighted a number of different vulnerabilities our current society has to reliability failures.

Safety: With an increase of computation in automobiles, airplanes, traffic light systems, and medical systems, the need for high-reliability systems will increase. While, in many cases, human fatalities decrease with increased computer automation, a reliability failure in these systems could

have drastic impact. For example, computer-aided braking can apply proper braking in icy conditions faster and better than a human can, but a reliability failure in the brakes could result in an accident. Therefore, continued implementation of computation in life-critical areas must be done with increased reliability requirements in order to decrease risk to human life.

Economic: The growth of the world-wide economy and well being has been fueled by cheaper and more powerful computations that enable greater automation and new services and products. This growth, in turn, has been fueled by Moore's Law scaling. Integrating reliability and variation management into designs is essential to allowing us to continue to extract size, cost, and energy benefits from scaled computations.

Energy: Energy consumption promises to be a limitation to our capabilities, our economy, and our environmental impact. Continued reduction in the amount of energy consumed per computation remains an important tool in expanding our computing capacity and taming our energy consumption. In turn, energy-efficient computation will allow us to optimize the use of scarce resources in other industries and activities by increasing efficiency and reducing waste.

Ultra-reliable Systems: Computation plays an increasing role in our infrastructure, including utilities and financial systems. As computation is a necessary component of our fast-paced financial systems, our economic systems have always needed to rely upon ultra-reliable, high-availability systems that can guarantee the security of the world economy. For our utility infrastructure increased use of computation (Smart Grids) will make it necessary to guarantee that computational errors never undermine the power grid. Unless we systematically address reliability issues, these systems will be hit by the double-whammy of increasing device count and decreasing device reliability.

Harsh Environments: Automated computations expand our reach and survivability into harsh environments, such as space, high altitude, or extreme temperatures. However, these environments increase the upset and wear rates for devices, an effect that is further magnified as devices scale down in size. We must be able to scale our reliability solutions to these more extreme environmental characteristics and do so with modest incremental effort on top of mainstream designs.

Security: As more of our interactions are managed and enhanced by computer mediation, it becomes increasingly critical that these systems be robust against deliberate subversion attempts. Guaranteeing that a system cannot be penetrated is a difficult task even when we assume the devices and components work perfectly. Misbehaving devices violate key assumptions and create a myriad of new attack vectors against our systems. For example, researchers have already identified ways in which soft errors can be used to defeat cryptographic systems [134, 116, 94] and software isolation layers [54].

We need to change our computational infrastructure now, so that we can continue our economic growth, quality-of-life improvements, and expansion of technological capabilities with trustworthy computation.

2.8 Why Government Leadership?

Government leadership is essential. The work necessary to achieve cross-layer reliable systems crosses the entire computing system ecosystem from integrated circuits to software applications (See Section 11). Therefore, no one vendor or research laboratory will be able to effect change by themselves. Wide-scale cooperation across specialties and organizations is necessary to revolutionize computing systems in this manner, otherwise the community will be facing yet another

stop-gap revision that will only postpone these problems for a few more years! The government will also likely have to intercede should a reliability issues affect the nation's economic, infrastructure and security systems. The large costs and risks associated with unreliable electronics are mostly external, impacting society at large rather than electronics producers, suggesting that pure market forces alone will not drive desirable and adequate solutions. Therefore, government input and help is necessary in guaranteeing reliable computation in these sectors.

There is also a strong economic argument for why this research needs to happen here in the United States. Over the last several decades, the US has been a primary contributor to the electronics revolution and has reaped substantial economic benefits from the products our inventors have developed. The research outlined above will allow US companies to sustain Moore's Law for at least another decade, will provide them with the technologies required to create innovative, reliable, products that will bolster our economy, and will create jobs in the US technology market, helping to fight the trend towards outsourcing the design and manufacture of electronics overseas.

2.9 Summary

Moore's Law feature-size scaling has been an economic and capability engine fueling wealth creation and quality-of-life improvements for the past 40 years. We are now moving into a qualitatively new regime where device energy will be the dominant limitation on the exploitation of additional capacity and where devices are inherently unpredictable and unreliable. In this new regime our old solutions to reliability no longer make sense and will lead to an early end to the benefits of scaling. However, by exploiting information rather than energy to tolerate errors at higher levels in our system stack, we can productively exploit these smaller technologies to continue reducing energy while ensuring high system-level safety. In the remaining sections, we elaborate on the challenge, how it impacts various industries, and the nature of the research required to address this challenge.

3 Examples and Illustrative Scenarios

During the study, the team developed a number of strawman designs for cross-layer resilient systems. In these section, we outline some of these designs as examples of how cross-layer reliability might be implemented. Much of this work was originally published in greater detail in [36].

3.1 Resilience Tasks

Previous work on reliability has tended to focus on developing mechanisms to address particular causes of errors or variation, such as single-event upsets and negative-bias temperature instability. This approach can be effective when a small number of physical mechanisms cause the vast majority of errors, but is becoming increasingly expensive as the number of phenomena that cause errors, variation, and aging increase, and as increasing error rates force increases in the number of structures on a chip that must be protected. In particular, the cost of verifying and testing all of the reliability mechanisms used in a design is becoming a significant issue.

When designing cross-layer approaches to reliability, it is often more useful to think of implementing reliability/resilience using a set of *tasks*, each of which handles a different aspect of a system's response to errors or variation. These resilience tasks can be thought of as steps that the system follows to handle a particular error or variation, although they may not occur sequentially. From our initial discussions, we defined five resilience tasks:

1. **Detection:** Determining that an error has occurred (i.e., that some fault has caused one or more bits in the computation to differ from their correct value).

2. **Diagnosis:** Characterizing the system's state to locate the causes of errors, determine how the system is changing over time, and predict the onset of aging-induced faults.

3. **Reconfiguration:** Changing the state of the system to prevent an error from recurring and/or to prevent variation or aging from causing errors.

4. **Recovery:** Ensuring that an error does not propagate to user-visible results, for example by rolling back an application and re-trying one or more failed operations.

5. **Adaptation:** Re-optimizing the system to provide the best possible performance/power given the changes to the system state made by the reconfiguration task.

Figure 17 illustrates how resilience might be distributed across the system stack using this task list. Circles indicate layers that participate in a particular task, while arrowheads indicate the direction of information flow in the task. This example illustrates a system where applications are able to participate in reliability/resilience, but are not required to, as might be found in a general-purpose computer that needs to be compatible with older software. Communication from the application to the rest of the system that is not required is shown in gray.

In this example, errors are detected at the circuit and architecture levels, and information about errors flows up to the operating system. A combination of OS-directed tests and sensors at the circuit and architecture levels continually diagnoses the state of the system to detect aging and variation. The operating system also controls recovery from errors, reconfiguration to prevent errors from recurring, and adaptation to changes in system state.

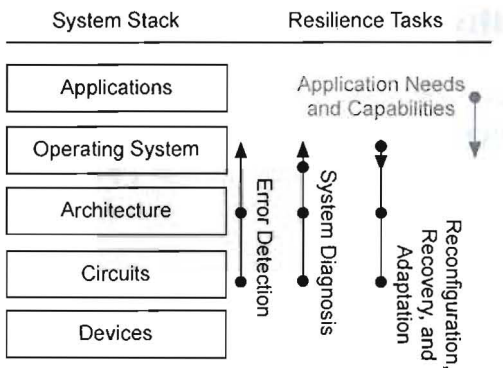


Figure 17: Distributing Resilience Across the System Stack

3.2 Cross-Layer Reliable Computing Systems

Figure 18 illustrates an example of how a cross-layer resilient general-purpose computing system (laptop, workstation, etc.) might be implemented. For the purposes of this discussion, we assume that the system is required to be backward-compatible with older software, and thus discuss scenarios where applications both are and are not involved in resilience. As shown in the figure, the computing system's hardware consists of a many-core CPU and off-chip DRAM. The operating system incorporates four software sub-systems that contribute to resiliency: an error handler routine, a resource map, which describes the current state of the system, a hardware configuration routine that controls the system's hardware, and a task scheduler, which takes the information in the resource map into account when scheduling tasks.

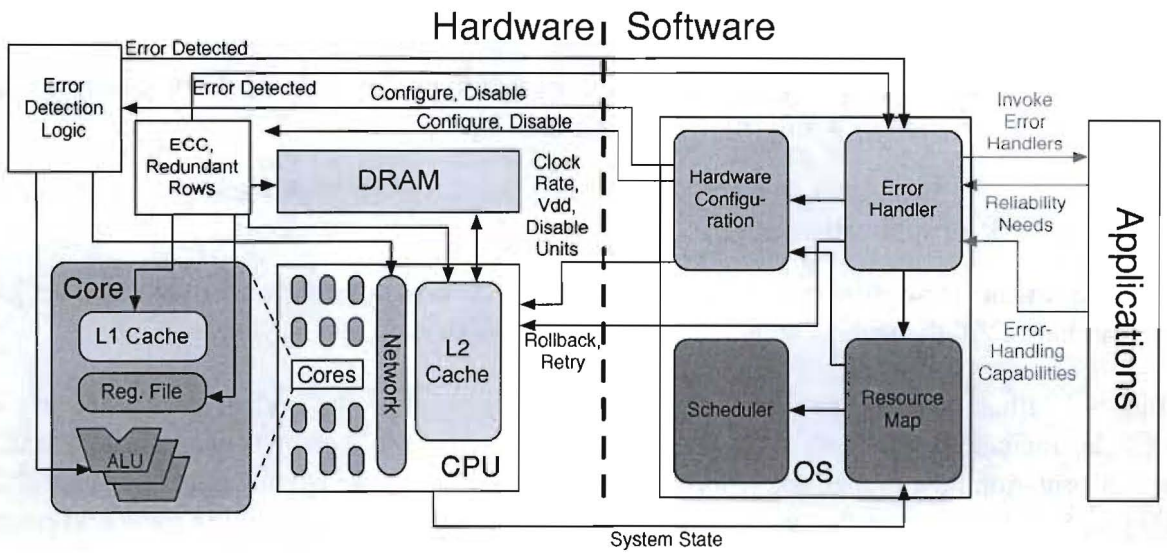


Figure 18: Example Cross-Layer Reliable Computing System

3.2.1 Detection

When applications are not involved in resilience, errors are detected at run-time by a set of low-cost hardware mechanisms, such as ECC codes on memories and parity/residue codes on computations. When an error is detected, the hardware signals an error handler in the operating system, which directs the recovery, reconfiguration, and adaptation tasks.

Resilience-aware software can significantly improve error detection rates and/or reduce the amount of hardware required to detect errors. Algorithmically, checking the results of many computations potentially takes significantly less time than is required for the computation itself (See Section 5.4), although exploiting this behavior typically requires programmers to invest significant effort. Alternatively, some algorithms, such as matrix operations, can be modified to operate on data structures that incorporate checksums or other redundancy, allowing them to detect and/or correct errors [64].

Compiler-based techniques can detect many errors in software by observing common error symptoms or violations of invariants [131][111], and can insert redundant instructions to detect errors [90][91][105]. Combining a resilience-aware compiler with appropriate hardware support can also effectively detect control-flow errors, such as branches taking the wrong path, which can be hard to detect in hardware [74] [79][129][53][93].

One challenge in application-based resilience is that software development is often a multi-layered, multi-organizational endeavor. In particular, applications increasingly incorporate code from libraries, meaning that an application may be a combination of resilience-aware and non-resilience-aware code. Because of this combination of user-created and pre-compiled libraries, techniques for using non-resilience-aware libraries in resilience-aware applications or increasing the reliability of pre-compiled, non-resilience aware libraries will be extremely useful. Techniques to isolate software blocks to contain the effects of bugs, errors, and security violations will also be useful in heterogeneous software environments to prevent pre-compiled libraries from decreasing the reliability of resilience-aware applications.

3.2.2 Diagnosis

In a cross-layer resilient system, error and system diagnosis are performed by a combination of hardware and software mechanisms. Temperature and supply voltage sensors can provide valuable information about the short-term state of the system, while delay sensors on logic paths [47][13][31], diagnostic circuits [22], and periodic hardware or software self-test [120][38][92] can help to diagnose longer-term variation and aging. Regardless of the set of diagnosis mechanisms a system implements, their outputs are sent to the OS and used to update its resource map with information on which units in the system are operating correctly and allowable clock rate/supply voltage combinations for each unit. System diagnosis can also help predict permanent errors before they occur by noting changes in transient fault rates and/or transistor behavior [73].

Because diagnosis is intimately tied to the system's hardware, it is difficult for application-level software to contribute significantly to this task in general-purpose computing systems. Single-purpose or embedded systems might choose to integrate the software-based test techniques mentioned above into applications to reduce the amount of system software they require, and applications that incorporate error detection techniques can inform the OS of the number of errors they detect to help it diagnose slow-onset errors.

3.2.3 Reconfiguration

There are two aspects to reconfiguration in a cross-layer resilient system: selecting a set of resilience mechanisms to use that provide sufficient reliability while maximizing performance/Watt, and adjusting the operating points and capabilities of different hardware units in response to faults, aging, and variation. Resilience-aware applications can contribute significantly to the first aspect of reconfiguration by informing the operating system of their capabilities, allowing it to disable hardware mechanisms that are not needed. Even when executing resilience-unaware applications, there is potential to select resilience mechanisms based on the system's needs, for example by selecting the amount of error-correction used in the memory or registers based on the observed error rate [138].

Techniques to reconfigure a system to account for faulty or aging hardware vary significantly with the type of hardware being reconfigured. The regular structure of memory arrays makes it possible to disable small regions of an array in response to faults [133]. Network techniques to tolerate failures are relatively well-studied, and efforts are exploring ways to apply these techniques to on-chip networks [46].

Reconfiguring execution resources involves trade-offs between the overhead of the reconfiguration mechanisms and the amount of hardware a given fault/variation can affect. Given the consequences of disabling an entire core in current-generation chips, a number of efforts have explored techniques to tolerate faults in execution units by exploiting redundant hardware in superscalar cores [115][29], combining multiple faulty cores into a single "virtual" core [108] or by relocating tasks if they try to use a faulty unit on a given core [97]. As the number of cores per chip increases, it may become more attractive to treat cores as atomic units, adjusting their clock frequency and supply voltage and/or disabling them in response to faults as long as the CPU provides mechanisms to isolate faulty cores from the rest of the system [9] and migrate tasks to healthy cores [103].

Like diagnosis, reconfiguration is a sufficiently hardware-specific task that it is difficult for applications to contribute to reconfiguration, although compiler-based reconfiguration schemes that replace instructions that require faulty hardware with software emulation [78] have been proposed.

3.2.4 Recovery

Error recovery can benefit significantly from a multi-layer approach, as single-layer recovery techniques, such as triple modular redundancy or application checkpointing [96], have very high costs. Much of the difficulty in error recovery comes from the wide variance in the delay between the time when a fault occurs and the time that the resulting error(s) are detected. When errors are detected quickly, they have little ability to impact state, and low-overhead recovery techniques, such as squashing instructions in the pipeline, are possible.

However, some errors, such as network errors and multi-bit memory errors, are inherently difficult to detect quickly and therefore require checkpointing and rollback or other more-powerful error recovery techniques, which are still very much an open area of research. A number of projects have investigated techniques that leverage the redundant storage of data in cache hierarchies to provide low-overhead checkpointing [65][98][123]. While they can significantly reduce checkpointing overhead, these techniques have limited ability to guarantee that a checkpoint will be kept alive for a specific number of cycles, because movement of data into and out of the caches is determined by the application's memory accesses.

Integrating applications into the recovery task has the potential to greatly reduce recovery costs

by only checkpointing the data that is necessary to roll back the application. Simply having the application determine when checkpoints should be taken can greatly reduce checkpointing overhead [96], while applications that provide their own checkpointing code can see even greater improvements [51]. A more general approach to application-level redundancy might involve the use of side-effect-free programming styles, in which functions are not allowed to modify any data other than their return values, making it possible to re-execute them when errors occur, or transaction-based programming, which allows computations to be aborted and restarted if an error is detected before the computation commits its results.

Transactional computing has always been popular in financial computing and database computing, and has become increasingly popular in web and enterprise applications. In enterprise applications, the transactional model is already being used to support micro-reboots to tolerate application corruption [35]. Transactions are designed to abort and restart a computation and can use this capability to recover from hardware corruption as well. As multicore architectures pull parallel programming into more mainstream usage, there has been significant work on transactional memory models [56, 24]. This trend provides the hardware support for efficient transactions in programs, encouraging the further migration of applications to the transactional model.

3.2.5 Adaptation

The adaptation task is responsible for deciding how to allocate tasks and power to the different units in the system in order to maximize overall performance without exceeding the power budget or other constraints. In our strawman system, the operating system handles adaptation, using the output from the diagnosis task and its knowledge of which applications are running to set the clock rate and supply voltage of each unit in the CPU and to determine the mapping of tasks to units.

At the application level, many of the same techniques used in implementing high-performance parallel programs also increase an application's ability to adapt to changing hardware. For example, dynamic load-balancing techniques can tolerate variations in thread run-time caused by either varying amounts of work in each sub-task or by cores operating at different clock rates. Similarly, applications that can increase or decrease the number of threads they use can adapt to changes in the number of cores available to them, regardless of whether those changes are caused by hardware faults or by other applications starting or stopping.

One challenge in adaptation is that it is currently very difficult for the system to predict how an application's performance will scale with parallelism, making it hard to decide whether it would be better to run the application on a small number of high-frequency cores or on a larger number of lower-frequency cores. A cross-layer system that provided an API for applications to pass predictions about their performance scaling to the operating system could significantly improve the OS' ability to allocate resources.

3.3 Cross-Layer Reliable Systems-on-Chip

In contrast to microprocessors, which are typically designed by a single company or organization, systems-on-chip rely heavily on intellectual property (IP) blocks, which are often licensed from organizations different than the one designing the system. This situation poses two additional challenges to reliability over and above those found in microprocessors. The first challenge is verification. In order to protect their designs, organizations often distribute IP blocks in netlist or encrypted formats that make it difficult for system designers to understand the internal workings of

the IP block. Encrypted IP cores makes it harder to verify and test SoCs, leading to increased rates of bug escapes into products.

The second challenge is that IP blocks are often re-used in multiple designs, which may have very different reliability requirements. An IP block designed to meet the reliability needs of an aircraft control system, for example, would be over-designed and inefficient if used in a portable audio player. Without the ability to pass information about the needs of the end application to IP blocks, it is very hard to design blocks that can meet the needs of a wide range of applications.

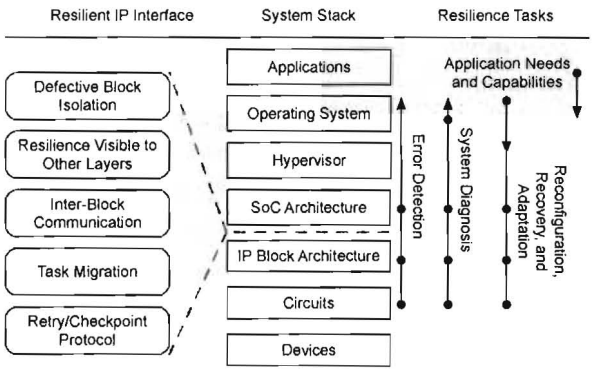


Figure 19: Resilient IP Interface

To address these challenges, the study group proposed the development of a resilient IP interface for systems-on-chip, as shown in Figure 19. The SoC industry currently defines a number of standard interfaces for IP blocks in order to simplify integration of IP blocks into a system. The resilient IP interface would extend these interfaces to provide mechanisms to isolate defective blocks from the rest of the system, make resilience needs and capabilities visible to other layers, provide inter-block communication, and support a retry/checkpoint protocol.

Using this resilient IP interface, a cross-layer resilient system-on-chip could be implemented as shown in Figure 20. The hardware portion of the system consists of an SoC, possibly with external DRAM (not shown). The SoC is made up of microprocessor cores, RAM arrays, and IP blocks. The cores and RAM arrays are assumed to have similar error detection and correction hardware to the ones in the computing example, but these are not shown to avoid cluttering the figure. The operating system contains the same set of services to support reliability as in the computing system, with the addition that the hardware configuration service must now also pass information about the system's reliability needs to the IP blocks.

The microprocessor cores communicate with the RAM arrays and IP blocks over the resilient IP interface, using an ACK/NACK protocol in which the requesting unit is required to retain the data required to re-generate any request until the unit that receives the request acknowledges that it has successfully completed the request. This interface provides a simple mechanism to correct transient errors by re-trying the computation that sustained the error. If an IP block sustains a permanent error, the SoC can migrate any tasks assigned to that block to another block that provides the same functionality, or can fall back to executing the task in software, albeit at lower performance.

The request/retry protocol also provides a mechanism to tolerate bugs that escaped the validation process by falling back to software when a bug produces an erroneous result that the system can detect. Bugs that are not caught in validation tend to manifest only in uncommon circumstances, such as particular input values or particular sequences of events, since bugs that manifest

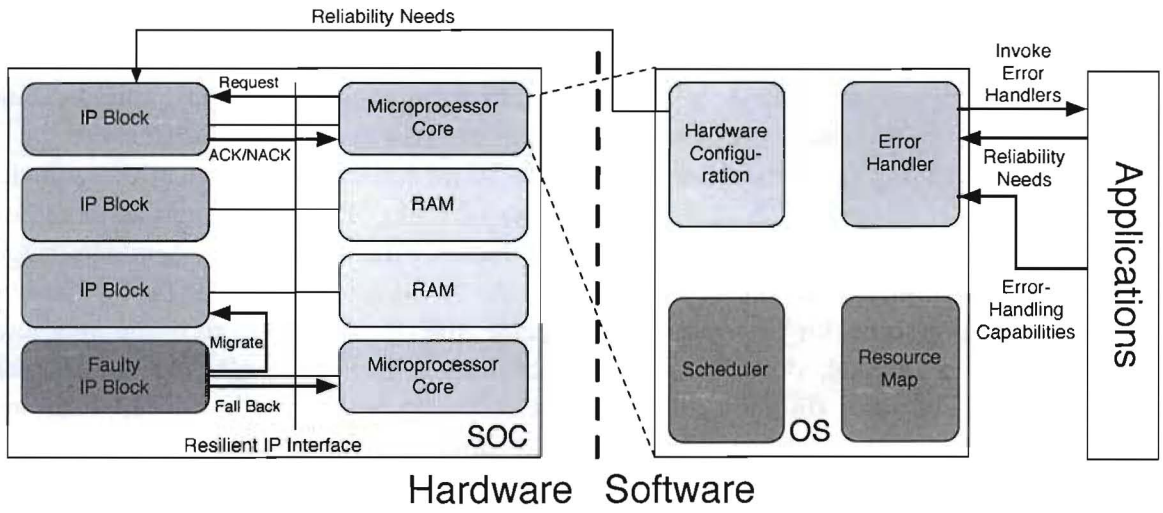


Figure 20: Example Cross-Layer Reliable System-on-Chip

frequently have a high probability of being found. For example, a system might take the view that the first failure of a given request is likely a transient error and re-try the request on the same unit. If the request fails twice, it might assume that a bug has been uncovered and fall back to a software implementation of the computation. Frequent repeated failures indicate that the IP block has aged or developed a permanent fault, triggering reconfiguration to disable the block, decrease its clock rate, or increase its supply voltage.

Systems that rely on this ability to tolerate buggy hardware should provide error-detection mechanisms that check the validity of larger sequences of computations, as opposed to individual instructions (Section 5.4), since hardware bugs may produce results that are self-consistent but incorrect. For example, a bug that caused an ALU to occasionally subtract instead of adding would not be detected by parity checks on the ALU's output.

3.4 Early Support for Reliable and Adaptable Application Software, Operating Systems, and Middleware

In an ideal cross-layer resilient system, every level of the system stack is involved in reliability and resilience. However, the longer design cycles and market turnover times of computing hardware make it interesting to consider how software-only cross-layer techniques could improve reliability in the absence of reliability-aware hardware. In the absence of more reliable hardware, changes to the application, operating system, and middleware layers can be useful in helping software applications survive minor hardware failures and guarantee correct operation. Furthermore, software-level changes might provide a flexible framework for adapting systems designed for a benign environment to be useful in a harsher environment.

Even without hardware support, applications that implement the self-checking techniques discussed in Section 5.4 may be able to detect errors, although typically at the cost of higher error detection latency than hardware-based detection schemes. Developing a better understanding of the theory of lightweight application checks will both help software developers know when their applications might be able to check their own work effectively and reduce the amount of work it

takes to add self-checking to applications.

The operating system and middleware levels of the system stack can do some amount of system diagnosis and reconfiguration. Both the operating system and the middleware have more resources for storing and processing error information than hardware-based approaches. The systems can store information regarding failed assertions from the software and state-of-health information from the hardware, which can be used to spot the indicators of failing hardware, such as increases in transient error rates. Once failing hardware has been located, the operating system can trigger a hardware reconfiguration, a different scrub rate, adapt the scheduler to avoid the failing hardware, or reroute computation to healthy hardware. In this sense, the operating system or the middleware can provide the necessary methodology to adapt the hardware, even if the hardware is not adaptable.

Software-based checkpointing and rollback techniques can be integrated into applications to allow recovery from errors without hardware support, although these schemes typically impose significant overhead. Techniques for state isolation [35], and transactional memory [118] might reduce checkpointing overheads. Alternately, applications that can correct errors by re-executing functions or through arithmetic techniques could avoid the overhead of checkpointing completely.

One weakness of these recovery techniques is that they require modification of the application, and thus have limited applicability to legacy software. Techniques to automate state preservation such as self-tuning monitor routines that observe a program's execution and determine the best times to halt it and checkpoint state, could greatly expand the set of applications that can gracefully recover from system crashes without hardware support. Legacy applications are a significant challenge for software-only cross-layer reliability, and techniques to increase the reliability of such applications are expected to be a significant area of research.

4 Challenge Problems and Areas of Pain

During our study, we formed focus groups in five key areas:

1. Consumer electronics: cell phones, laptops, etc.
2. Aerospace: airplanes and satellites
3. Large-Scale Systems: supercomputers and data centers
4. Life-Critical Systems: automotive and medical
5. Infrastructure: communications, power grid, etc.

In Section 4.1, we summarize the key challenges identified by each of the focus groups. (See Appendix E for full reports from each of these focus groups.) In Section 4.2, we identify a number of common challenges faced by designers in all of the groups and some possible directions to address these challenges. Many of the challenges and opportunities are characterized by the current need for each layer to operate without information potentially available at different layers in the system hierarchy stack, underscoring the need for cross-layer information sharing. Table 1 and annotations in both sections show how the focus group problems feed into the common challenges identified.

4.1 Challenge Problems by Constituency Group

4.1.1 Consumer Electronics (CE)

1. Conventional reliability mitigation techniques could negate the traditional benefits of scaling, suggesting a need for new, lower-overhead techniques to address the growing reliability challenge. (Contributes to 4.2.3)

2. Conventional margining in the face of growing variation could lead to increases in energy per operation, suggestion a need to develop new techniques to reduce energy per operation while retaining reliability. (Contributes to 4.2.4, 4.2.6)

3. Device-level wear effects could shorten component lifetime below acceptable levels, suggesting a need for techniques to maintain or extend component and system lifetimes in face of increasing wear. (Contributes to 4.2.4, 4.2.6)

4. Since few markets can demand unique components, but many applications and markets have different reliability needs, there is a need to find new techniques to economically address demand for components with different reliability needs. (Contributes to 4.2.1, 4.2.3, 4.2.5)

5. Reliability, area, delay, energy, and thermal issues as well as multi-layer solutions present an increasingly complex, multidimensional design space that may offer better design points, but cannot be explored adequately with reasonable time and engineering budgets given current techniques. (Contributes to 4.2.7)

4.1.2 Aerospace (AS)

1. Many systems are currently underpowered compared to current and projected computational needs.

(a) The widening gap between mil/aero and commercial parts leave many aerospace systems underpowered. (Contributes to 4.2.1)

(b) Design for worst-case environments sacrifices much needed computational capabilities. (Contributes to 4.2.1, 4.2.2)

(c) The complexity of the multidimensional optimization problem coupled with limited human time leads to suboptimal designs, sacrificing computational capabilities. (Contributes to 4.2.7)

(d) System reliability is considered intractable by conventional approaches, leading to a focus on part reliability that results in slow, heavy, power-hungry, and expensive solutions. (Contributes to 4.2.3, 4.2.7)

(e) For military and national security missions, guaranteeing that parts are not counterfeit and have not been tampered with has become difficult, preventing these systems from using the most advanced technologies and components.

2. Current approaches demand expensive testing that contributes to the lag between silicon development and exploitation and severely limits the number and kinds of components that can be used.

   (a) This challenge arises, in part, from the fact that low volumes and the enormous expense of chip design makes it impractical to design all parts uniquely for aerospace systems. One of the costs of commercially-available parts is the lack of adequate information about the design of the components, forcing testing to treat them as black boxes to reverse-engineer their potential weaknesses and vulnerabilities. (Contributes to 4.2.1, 4.2.5)

3. System lifetimes and development times are currently large compared to changes in the political environment and scientific needs, leading to systems that are poorly matched to current needs. (Contributes to 4.2.1, 4.2.3, 4.2.6)

4. Modern and future device technologies are likely to exhibit increased aging effects and in-system failure of devices. Increased aging effects could lead to unacceptably short system lifetimes. (Contributes to 4.2.4, 4.2.6)

5. Analog sensors, discrete components, and passive components are not reliable enough. (Contributes to 4.2.8)

4.1.3 Large-Scale Systems (LS)

1. Using conventional approaches, the energy overhead to achieve adequate reliability for large-scale systems is too large. Energy and reliability concerns could become impediments to further scaling up in computational power.

   (a) Worst-case assumptions about component susceptibility, fault propagation, and fault impact lead to considerable energy expense. (Contributes to 4.2.2)

   (b) The demand that reliability be handled at the system level with no guidance about the reliability needs of the application or application-level options for mitigation is an excessive and growing energy and performance expense. (Contributes to 4.2.3)

(c) Due to the relatively low volumes for large-scale systems and the high cost of custom design, these systems must be built from commodity hardware that is not designed to support systems of this scale (Contributes to 4.2.1 and 4.2.5)

2. There is no standard way to assess and validate the reliability of a system for the user and potential customer. (Contributes to 4.2.7)

4.1.4 Life-Critical Systems (LC)

1. Fault-tolerance/resiliency design choices are difficult to implement in a timely and efficient manner. (Contributes to 4.2.7)

2. Error correlation between device physics and architectural effect is poorly characterized and categorized. (Contributes to 4.2.7)

3. Sensors, discrete components, and passives are essential elements for most safety-critical systems and can limit overall system reliability, but are generally ignored in the literature of resilient design. (Contributes to 4.2.8)

4. It is often unclear how or where to make tradeoffs between the electronic components of a safety-critical system and the application in which it is embedded when many resiliency techniques are available and potentially interact. (Contributes to 4.2.7)

5. Translation from concepts to practice is hindered by the lack of a standard model for describing, comparing, and composing resilience techniques. (Contributes to 4.2.7, 4.2.5)

6. Exploring and characterizing resilient/fault-tolerant design techniques and their efficacy in simulation and/or hardware is time-consuming and expensive. (Contributes to 4.2.7)

7. It is currently difficult or impractical to integrate safety standard protocols and certifications into design flows.

4.1.5 Infrastructure (IS)

1. Affordable techniques are needed to increase the availability of increasingly large and distributed infrastructures systems. (Contributes to 4.2.4 4.2.6, 4.2.3)

2. Infrastructure systems cannot afford to develop large numbers of custom components for their applications and systems or to maintain processes and technologies unique from other market segments. (Contributes to 4.2.1, 4.2.5)

3. While compute costs do not currently dominate, it, nonetheless, remains true that providing complete, guaranteed never-fail service is prohibitively expensive.

4. Human service costs must decrease despite increasing component complexity, increasing system size, and increasing distribution of components. Advanced technology that may see earlier wear-out exacerbates this challenge. (Contributes to 4.2.4, 4.2.6)

5. Analog sensors, discrete components, and passive components are not reliable enough. (Contributes to 4.2.8)

4.2 Challenge Roundup

We can trace the root causes of many of the challenges and compromises seen in today's system designs to the need to design and operate specific system layers without key information. This information gap underscores the potential opportunities for cross-layer information sharing to address power and reliability challenges. In this section, we see that many of the individual challenges identified above are manifestations of information deficiencies. Table 1 shows how these common challenges underlie the individual challenges identified by the constituency groups.

4.2.1 Late-Bound Information

Environment, energy demands, deployed system context, and even technology noise and maturity are all late-bound, often being unknown during design and perhaps not known until deployment. The lack of information leads to both over-design for most scenarios (AS1b) and limited ability to use the components in more demanding scenarios (*e.g.* higher defect and fault rate than anticipated, larger environmental variations, more critical deployment contexts—AS1a, LS1c). This situation motivates the design of components and systems with modes and configuration options that allow higher layers in the system to tune what each component spends on reliability once late-bound information becomes known. This tuning can change with missions (AS3) and environment (AS1b). These modes will allow commercial devices to enhance yield or operate at extremely low energy levels [132] while also making the same parts more useful in larger scale systems or harsher environments (CE4, AS2a, LS1c, IS2).

4.2.2 Instantaneous Operational Information

Varying demands, workloads, environment and uncertainty about the environment even in a single system means that margins and mitigation techniques designed for the worst-case possible scenario are over-design for most operating hours [34] (AS1b, LS1a, LS2). This problem suggests a need for systems that can monitor their environment to extract the missing information and adapt to exploit this information about their situational needs.

4.2.3 Information about Application Requirements

Worst-case design for a platform independent of application needs is too expensive (CE1, LS1b, AS1d, AS3, IS1). Over-design arises when we demand that the platform, such as a supercomputer, provide a fixed, minimum level of reliability with no information about the tasks that are running on it. Similarly, worst-case design for uncommon, but potentially avoidable, worst-case scenarios is also a large, unnecessary cost [102]. This challenge motivates cross-layer, application-aware solutions. These solutions may include models and middleware that allow the application to communicate requirements and opportunities to the platform and that support management of operational and implementation aspects of an application mapping to a particular platform.

4.2.4 Information about Health of Components

Nanoscale components see large variation such that each component is different (CE2). Furthermore, components wear out over time (CE3, AS4, IS4). Systems that operate only based *a priori* information must margin for the worst-case elements across large systems and production runs,

Industry Challenge	Common Challenges							
	Late-Bound Information	Instantaneous Operational Information	Info. Application Requirements	Info. Health of Components	Info. Comp. Heterogeneous Suppliers	Granularity of Adaptation and Repair	Incomplete Info. on Reliability Weaknesses	Analog and Passive Elements
Consumer Electronics								
1. transient			•					
2. margin				•		•		
3. wear				•		•		
4. different needs	•		•		•			
5. multidimensional opt.							•	
Aerospace								
1. underpowered	•	•	•				•	
2. expensive testing	•				•			
3. lifetime >> changes	•		•			•		
4. aging				•		•		
5. analog/discrete reliability								•
Large Scale								
1. overhead too high	•	•	•		•			
2. reliability assessment							•	
Life Critical								
1. TTM Pressure							•	
2. poor understanding							•	
3. analog/discrete reliability								•
4. evaluating tradeoffs							•	
5. standard model					•		•	
6. selecting techniques							•	
7. standards integration								
Infrastructure								
1. inexpensive availability			•	•		•		
2. commodity components	•				•			
3. never-fail expensive								
4. decrease service costs				•		•		
5. analog/discrete reliability								•

Table 1: Common Challenges

leading to excessive margins and mitigation overhead for most devices (CE2, IS1). This problem can also lead to lower reliability and higher vulnerability for devices that exceed the predictions made at design time.

4.2.5 Information about Capabilities of Components from Heterogeneous Suppliers

Fully custom or unique construction of all components is not viable for any company, industry, or government agency (CE4, A1a, LS1c, IS2). Some domains see more acute versions of this problem, but no domain can afford the investment in time, manpower, and manufacturing costs to develop custom components for most or all of the parts used by their applications and systems. This scenario motivates the need for interfaces, metrics, benchmarks, and tools to perform composition, analysis, optimization, and validation of separately sourced (sub)components. System solutions must be cross-layer, with higher layers conveying context to lower layers and lower-layers communicating capabilities and health to upper layers (CE3, AS4, LC5). Lack of information drives inefficient and conservative use of components.

4.2.6 Granularity of Information Exploitation—Adaptation and Repair

In the past, the time between permanent faults was large enough and the functionality provided by each chip small enough that it was feasible to discard chips that developed even one defect and to manually replace field-replaceable units as they failed. However, as defect rates, variation rates, and the capacity of individual chips increase, it is no longer viable for humans to be in the repair and reconfiguration loop (temporal granularity) or to demand replacement of entire chips or boards in response to a single failure (spatial granularity). Furthermore, in order to exploit the additional information determined during system operation to improve reliability, availability, lifetime, and mission relevance and to decrease energy, we must be able to adapt systems at an appropriate level of granularity (CE2, CE3, AS4, IS1, IS4). This will allow the system to (1) avoid, or treat uniquely, the small fraction of the design with excessive variation or wear (4.2.4), (2) deploy resources differently in response to instantaneous environmental conditions (4.2.2) or application requirements (4.2.3), and (3) customize for the system environment (4.2.1).

4.2.7 Incomplete Information on Component and System Reliability Weaknesses

Across the board, there is considerable conservative over-design. Time-to-market pressures and limits on human time (CE5, AS1c, LC1, LC4–6) coupled with a lack of automation drive the acceptance of large margins and safety factors at many layers of the design. In many cases, safety factors are not effectively applied, providing too much guarding on most cases and components (A1d) in order to get an adequate level on a subset. This problem arises from a lack of visibility into the real sources of weakness in the design. There is a need for system assessment methodologies (LS2, LC2) and tools to support better and more automated exploration of tradeoffs in the energy-delay-area-reliability-thermal-mechanical design space. This situation is true both for chip-level design of processors and ASICs and for system-level design of satellites, supercomputers, cell-phones, and pacemakers.

4.2.8 Analog and Passive Elements

Analog, discrete, and passive components can be the weak link in many mixed-signal systems (IS5, AS5, LC3). The body of knowledge and state of the art around system-level design and reliability management for these components is small and inadequate to the variability and reliability challenges now arising. System-level reliability must also incorporate passive and analog components into analysis and mitigation schemes. Solutions need to be able to monitor the health of these components (4.2.4), tune them when possible (4.2.1, 4.2.2, 4.2.6) and compensate for the failure of individual elements.

5 Science Questions

While reliability has had attention in the past, both the orders of magnitude growth in machine sizes and the anticipated orders of magnitude growth in fault, variation, and defect rates demand new solutions. As Nobel Laureate physicist Murray Gell-Mann says, "three orders of magnitude is a new science."[1]

The common causes of the challenges identified in the previous section suggest that we need to develop a better **scientific and engineering** understanding of (1) the role of information (value of exploitation, costs of operating without information), (2) sustainable techniques for sharing information across the multi-layer system stack that supports computations, and (3) architectures and techniques for exploiting information as it becomes available. We can frame this need more specifically into the following set of fundamental questions:

1. How do we design hardware and software organizations that are prepared for repair?
2. What is the right amount of error filtering at each level in the stack—from circuits through application software—and what are the best techniques for filtering?
3. How do we formulate, analyze, and manage multilevel trade-offs for fault mitigation that generalize the idea of hardware-software trade-offs, including the interfaces for cross-layer information sharing?
4. What is a theory and design pattern set for efficient, light-weight error checking that exploits the high-level properties of hardware architectures, software applications, and algorithms?
5. What is a theory and practical framework for expressing and reasoning about differential reliability, including both application needs and hardware/system organization to meet those needs?
6. How do we design scalable system solutions that can adapt to varying error rates and reliability demands?
7. How do we design components and systems that degrade gracefully and systems that are aware of their overall health?

Table 2 provides an overview of how these questions provide the underlying science and engineering foundation to address the challenges identified in Section 4.

[1] John Holland `http://www.spacedaily.com/news/robot-03b.html`

Science Question	Common Challenges							
	Late-Bound Information	Instantaneous Operational Information	Info. Application Requirements	Info. Health of Components	Info. Comp. Heterogeneous Suppliers	Granularity of Adaptation and Repair	Incomplete Info. on Reliability Weaknesses	Analog and Passive Elements
How do we design hardware and software organizations that are prepared for repair?						●		○
What is the right amount of error filtering at each level in the stack—from circuits through application software—and what are the best techniques for filtering?			●				○	
How do we formulate, analyze, and manage multilevel trade-offs for fault mitigation that generalize the idea of hardware-software trade-offs, including the interfaces for cross-layer information sharing?	●	●	●	●	●		●	○
What is a theory and design pattern set for efficient, light-weight error checking that exploits the high-level properties of hardware architectures, software applications and algorithms?			●				○	○
What is a theory and practical frameworks for expressing and reasoning about differential reliability, including both application needs and hardware/system organization to meet those needs?			●					
How do we design scalable system solutions that can adapt to varying error rates and reliability demands?	●	●	●	●		●	●	
How do we design components and systems that degrade gracefully and systems that are aware of their overall health?			●	●	●	●	●	

Key: ● = strong connection between question and challenge, ○ = weaker connection

Table 2: Relevance of Questions to Common Challenges

5.1 Repair

How do we design hardware and software organizations that are prepared for repair?

We expect to see increasing rates of in-system failure and aging in the future. Increasing chip capacity further means greater capacity in each field-replaceable unit. At the same time, we see increasing pressures to decrease the need for human intervention during maintenance and repair. All of this points to a growing demand for architectures that can self-repair in their deployed systems. While mission-critical systems have provided some capability for sparing at the system level, the new demands will reach into individual integrated circuits, impact **all** systems, and demand coordination of repair across components and subsystems designed and produced by different vendors.

5.1.1 Granularity

One key question to understand better is the granularity of repair versus the expected defect and wearout rates. For low defect rates, processor core sparing may be adequate. However, as burnout and wear effects increase, the optimum granularity will shrink. Finer-grained sparing may include pipeline stages [55], cache lines [132], programmable gates [70], and interconnect segments [137, 110]. In general, finer-grained resource substitution may incur higher delay and energy overheads for configurability, while coarser-grained sparing discards more capacity with each defect. Clever architectural innovations may be able to avoid or reduce this tradeoff and may be unifiable with solutions for scalable and adaptable fault tolerance (Section 5.6).

5.1.2 Interfaces

A key question here is how to make the repair capabilities available to the system.

1. What interfaces expose the repair capabilities?
2. How do we make it tractable to assemble systems that exploit the repair capabilities?

Perhaps components will need to provide device-driver-like mechanisms that hide the lowest-level repair details from the operating system while still exposing a reasonably powerful interface for the OS to direct repairs. The component device-driver may include its own support for low-level self-test and diagnosis. These interfaces will also be related to health monitoring (Section 5.7), because reliability management services in the OS will need to be able to estimate how likely it is that a given component will be able to make future repairs in order to estimate the component's remaining lifetime.

5.2 Filter

What is the right amount of error filtering at each level in the stack—from circuits through application software—and what are the best techniques for filtering?

Demanding that all errors be filtered at a particular level in the system stack imposes a very high cost and is, in general, not the optimal way to partition functionality across the layers. In memories, for example, we learned that it was not best to demand that devices never be upset. Instead, we use error-correction codes (ECC) at a higher level to catch errors that device hardening does not eliminate. Furthermore, we employ scrubbing at an even higher level to correct errors before they accumulate enough to be uncorrectable by the modest ECC employed. Similarly, Internet-level

network communication manages errors at multiple layers with lower layers allowed to make errors but attempting to filtering out classes of errors before passing them on to higher levels.

Can we develop a better understanding of how to best filter errors? This will likely include an understanding of:

- What classes of errors are most easily (with least energy) filtered at which levels?
- What error rate can each level reasonably handle?
- Can we characterize the error filtering that is already inherent in the layer? (*e.g.,* [83, 67, 124])
- While there is work measuring raw rates of error filtering, is this adequate, or do we need more sophisticated models to characterize layers and components to allow composition and more accurate prediction of system-level reliability? (*e.g.,* [72])
- Can we understand how to design to enhance and tune error filtering at various levels?
- Can we develop a framework for deciding how much energy to expend at each layer to enhance its filtering?

5.3 Multi-layer Interfaces and Cooperation

How do we formulate, analyze, and manage multilevel trade-offs for fault mitigation that generalize the idea of hardware-software trade-offs, including the interfaces for cross-layer information sharing?

Sharing information and coordinating configurations and actions across layers emerged as one of the clear areas of need and opportunity in the study. While it is clear this needs to happen, it is also clear there are many issues to understand and resolve to determine the appropriate interfaces and the best ways to enable this optimization.

1. *What should new contracts and interfaces look like?* While there is agreement that the traditional layer interfaces need to change, there is also agreement that interfaces are necessary and enabling. Consequently, the question is how the semantics and signaling between the layers needs to evolve, and whether or not new layers are needed or layering should be structured in different ways.
2. *What information should be sent to higher levels in the stack?* (*e.g.,* rate of correction in ECCs, rate of low-level rollbacks) How do we control or tune information flow? While there is agreement that the higher stack levels need more information, it is also clear they could be overwhelmed with information they cannot use. For an interface, we would hope to enable a broad set of optimizations, including ones we have not yet conceived.
3. *How should lower-level events and effects be coerced into a smaller set that is recognized by higher level software?* What is the appropriate size and composition of this set that adequately distinguishes events that higher levels should treat differently without creating an unmanageable number of cases for higher levels to manage? Applications may be willing to add some support for hardware failures (*e.g.,* [50]), but the burden of handling all conceivable failures uniquely is too high. One solution is to coerce a large set of failures into a single type that the software can handle (*e.g.,* [121]). This solution is used profitably in the formulation of networking and distributed systems (*e.g.,* fail-stop coerces a whole class of misbehaviors into a machine ceasing before causing any damage [19], lost/discarded packets simplifies all the problems that may go wrong in network transmission into a single case for the end-point hosts to resolve).

4. *What controls on lower levels should be exposed and how?* Examples of low level controls include scrubbing rates for both memory and logic and the choice of ECC. Higher levels with a more global view on the application, context, and environment will typically have better perspective on the needs of lower level components. Consequently, a key part of cross-layer sharing will be enabling the higherlevels to tune sets of components to work together efficiently. Nonetheless, exposing controls without a good understanding of their semantics and interplay will not work either. This raises questions both about the right level of interface to expose and the appropriate layering. Perhaps the higher levels just have gross control (*e.g.,* spend more energy/area to increase reliability) and component-specific middleware translates higher level goals into lower-level configurations (*e.g.,* [84])?
5. *What information is it useful for higher levels to pass down, such as invariants, signatures, type properties? How do we select or control the information provided?* A wealth of information is currently discarded between the higher and lower levels of a system. Some of this information could be useful, but is currently not in a suitably usable format. However, it is also clear that much of it will not be useful or will only be useful in a suitably digested form.
6. *How do we evaluate and compose techniques across levels?* Cross-layer optimization, both offline during design and online during operation is a clear opportunity. How do we enable an automated system to understand the options that exist at each layer and coordinate amongst them? Alternately, how do we enable components to participate in distributed optimization algorithms? What algorithms can achieve quality results in these scenarios?
7. *How do we engineer and analyze adaptation and repair control loops across layers?* The functionality for the control loops for adaptation will, in general, be spread across the layers, integrating information provided by multiple levels and coordinating reconfigurations that can occur at each of the levels. We will need to find software and system architectures that facilitate composition of software and models from separate sources (*e.g.,* vendors) and facilitate analysis for robustness and efficiency.

5.4 Lightweight Checking

What is a theory and design pattern set for efficient, light-weight error checking that exploits the high-level properties of hardware architectures, software applications, and algorithms?

There are examples where applications allow reasonably simple end-to-end checks on their correctness (*e.g.,* factoring, finding a satisfying assignment to a SAT instance, ILP-solve, linear matrix solve). In these cases, it is much less expensive to perform the end-to-end application check than it is to check every intermediate step of the computation. For these cases, if fault rates are sufficiently low, it would be worthwhile to not spend energy on lower level checks or redundancy because the high-level checks will adequately protect the application from uncommon failures. It would be worthwhile to understand how many applications, or what portions of applications, can be checked this way and how to best express programs to allow systems to exploit these lightweight, end-to-end checks when they are available.

1. *What computational classes are lightweight checkable?*
   - *For what class of computations is it provably (asymptotically or a constant factor) less expensive to check (deterministically or probabilistically) a solution than to compute it?* It is well known that checking is cheaper than computing for some classes of hard

problems. For example, the definition of NP is that solutions are P-checkable. Can we establish this more broadly? For example, can we identify asymptotic differences between computing and checking within the class of polynomial-time computations? Can we establish bounds that include constants? If we cannot do this for the most general computations, can we establish bounds for specific classes of computations? (*e.g.,* log-checkable, α-checkable)

- *Can we enlarge this class by* extending *the output of the base computation so its results include a certificate to assist checking?* (*e.g.,* Extended GCD [23])

2. *How do we express checks in computations and optimize their use?* Assertions could be the most primitive form of embedded, application-specific checks, but it is likely we will want to capture more semantics to allow the system to decide which checks are needed. For example, when fault rates are low and a higher layer of the software has adequate end-to-end checks, it may be reasonable to omit checks by lower level components. Alternately, there may be different classes of checks that should be enabled/disabled based on the capabilities or fault-sensitivity structure of the hardware (*e.g.,* hardware that has modulo arithmetic checking may not need some arithmetic assertions that could be useful on architectures that do not have this hardware support).

5.5 Differential Reliability

What is a theory and practical framework for expressing and reasoning about differential reliability, including both application needs and hardware/system organization to meet those needs?

We do not need to protect all computations or even all pieces of a computation equally. At the very least, this gives us an opportunity to judiciously allocate the energy we spend on reliability (*e.g.,* where do we use higher voltages or components with larger feature sizes, where do we employ stronger levels of redundancy). However, this also opens up the opportunity to deliberately engineer systems where we minimize the portion of the system that needs high reliability, thereby minimizing where we must invest energy for reliability. While some systems have exploited differential reliability in *ad hoc* ways, this is an opportunity that has not been treated in the early theoretical work on reliable computation from unreliable parts (*e.g.,* [86, 95]) and has not been systematically developed and supported as part of design flows, languages, and tools. As such, this is a promising, untapped direction to further increase system reliability at low costs.

Components of developing a workable engineering approach include understanding:

1. *How do we express (or analyze or discover) and exploit allowable noise (error rates) from the description of a computation or piece of a computation?*
   - Many computations work with uncertain data (*e.g.,* noisy samples from the real world) and produce results that are probabilistic rather than absolute (*e.g.,* speech recognition). Many of these computations are already using algorithms that are prepared for errors at some level. As a result, modest amounts of noise may not impact the quality or acceptability of the results produced [62, 60, 119].
   - A convergent algorithm (*e.g.,* iterative matrix solve) may self-correct for large classes of errors.
2. *How do we express (or analyze or discover) and exploit noise sensitivity in the components of a computation?*

- While a probabilistic or convergent computation may be very tolerant to errors in its data calculations, the basic control that drives computation and decisions about whether or not to continue refining a calculation will typically be sensitive to errors. It may be useful to understand how much more protection these critical sections need to achieve a given level of task reliability. Done correctly, this gives us the most bang-for-buck for each incremental investment of energy toward reliability.
- Computations guarded by lightweight checks (Section 5.4) are not sensitive to errors, while their guards are sensitive.

5.6 Scalable Adaptive Reliability

How do we design scalable system solutions that can adapt to varying error rates and reliability demands?

In order to realize benefits from information, we will need to adapt computations and perhaps platforms to exploit late-bound information. This demands designs and architectures that can be scaled to provide just the right level of protection as error rates or reliability demands (Section 5.5) change. We need both a more complete and operational analytical basis for understanding how designs and implementations should scale and concrete architectures that can scale efficiently across ranges of reliability demands and error rates.

5.6.1 Motivation

Many effects suggest that error rates and reliability requirements will change such that we cannot design for a single error rate and reliability target without paying large overheads for worst-case design.

- Platforms may be employed in different environments (sea level versus high-altitude) that have different upset rates.
- Component may be employed in different roles (avionic flight system versus MP3-player).
- Aging effects may change upset rates (low upset rates when new, increasing upset rates as a system approaches end-of-life).
- There is considerable uncertainty about the reliability problem at many levels. While some phenomena are understood and amenable to lower-bound analysis, many new effects become significant at the nanometer scale, and it is likely we will not fully understand the full range of effects before we must deal with them (before we start manufacturing or fielding chips). Furthermore, the recent emergence of new materials in the manufacturing process, such as copper, high- and low-K dielectrics, and carbon nanotubes along with the prospect of new bottom-up assembly techniques exacerbates this effect as the community has less experience with any potential side effects these materials could cause.

5.6.2 Theory

On the theory side we need to expand our notion of area-time (*e.g.*, AT^2 [128, 112]) and energy-delay (*e.g.*, $E\tau^2$ [76, 80]) tradeoffs to include reliability as a function of error rates, performance, and energy. We would like to know the minimum achievable energy-delay-area-reliability surface for a particular computational task as a function of error rates. That is, at any point in time we

want to achieve some target reliability (*e.g.*, FIT=1). As a function of the upset rate (*e.g.*, every transistor fails with probability 10^{-20} on each cycle of operation), how much capacity (some combination of energy-delay-area) is required to achieve the target? Perhaps there is a theory similar to Shannon's Rate-Distortion curves [117] that captures the feasible tradeoffs between capacity and reliability? The need to incorporate the complexity of the computational task makes this is more difficult problem to capture and analyze than data communication. As with rate-distortion curves in communications, these limits will help us understand what is possible and what tradeoffs we should be trying to achieve with practical implementations.

5.6.3 Architecture

On the architecture side, we need to develop systems that are efficiently scalable and tunable to different upset rates and reliability targets. Assessment of an architecture may include:

1. How close can it come to the theoretical bounds?
2. Over what range of upsets and reliability targets can it scale in system?

5.7 Graceful Degradation

How do we design components and systems that degrade gracefully and systems that are aware of their overall health?

Ideally, we will design systems to tolerate the required upset rates and provide adequate reliability while performing their tasks. However, provisioning sufficient spare capacity to always provide full service for uncommon or unanticipated problems can be excessively expensive (See IS3 in Section 4.1.5). Consequently, we would like systems confronted with extreme scenarios to degrade gracefully rather than catastrophically. This demands that the system be aware of the health of its components and able to use that information to understand its overall health and capacity. As more capacity is either lost due to permanent errors and aging or must be dedicated to countering increased upset rates, the available capacity in the platform diminishes. When the available capacity decreases below the requirements of the computational task, how do we adapt to the reduce capacity while still providing acceptable services for critical, perhaps real-time, tasks? This may demand that we shed less-critical work or reduce the quality of the answers provided. Possible components include:

- How should we capture and communicate the criticality of tasks so the system can make intelligent decisions about where to invest its remaining capacity?
- How should we capture capacity-quality tradeoffs and embed them in implementations so the systems can back off gradually to lower quality results? This will involve communicating or exposing quality-capacity parameters to the operating system.
- How do systems determine and monitor the health of their components?
- How can the components and the system as a whole estimate and communicate their current reliability, capacity, and expected lifetime?
- What are the control algorithms needed to distribute capacity among the critical tasks?

6 Mission Impacts

Reliability directly impacts many key *national missions* and human life. Many national security missions, including energy security, nuclear security, and warfighter support, depend on reliable computation. On top of it, humans come into direct contact with computing systems through CyberPhysical computing devices, such as drive-by-wire technologies or medically-implanted devices. In this section we will dicuss a number of ways increased reliability in national mission and CyberPhysical computing systems are necessary.

6.1 Save Lives by Enabling More Powerful and Reliable Life-Critical Electronics

As discussed in Section 2.2, many industries are becoming increasingly reliant on integrated circuits in applications where failures could cost lives. Medical electronics and medically implantable devices, such as pacemakers, are becoming increasingly sophisticated, and threaten the patient's health if they malfunction. Similarly, more and more vehicles are incorporating control electronics in applications such as drive/fly-by-wire where failures can lead to deaths. Cross-layer reliability will allow more-reliable implementations of existing life-critical products and will also provide the performance and power efficiency required by future systems.

Medical Techology: Currently, one of the most commonly used medically implanted device is the pacemaker for regulating heart functionality. The initial designs and manufacturing for pacemakers happened in the 1950's [5] and have changed greatly over that time. During this time, the number of initial implants of pacemakers has grown steadily [21], and approximately 500,000 Americans have pacemakers. The largest group increasingly adopting pacemakers are above the age of 80 and 85% of all initial pacemaker implantations are done on people over 65 [21]. While designed to be ultra-low-power and reliable, these devices do fail. "Severe and accelerated battery depletion, manufacturers' advisories, and electronic or connector defects accounted for 13% of pulse generator removals. The proportion of pulse generators removed from service as a result of manufacturers' advisories, electronic failure, and housing defects were 4%, 2%, and 1%, respectively [59]." Given all of these circumstances, changing the pacemaker technology to track closely the commercial electronics market can be risky. While smaller, lower-power devices are always desirable, employing newer technology could increase the failure rate of the devices. Cross-layer reliability techniques can be employed to allow designers to find better solutions for balancing reliability with power efficiency.

In the future, reliable, ultra-low-power computing systems will enable a host of breakthroughs in medical technology, including personal genomics techniques that allow drugs to be tailored to an individual's biochemistry, sensing devices that help compensate for blindness, assistive technologies that allow the elderly to remain more independent, more life-like artificial limbs, and implantable devices that operate for years without failure or the need for recharging. As these devices are on the forefront of computing and medical research, these devices will likely need more computation than is currently available in portable medical devices. Reliability will need to be a key component in these systems, as computational malfunctions could put the person in danger. A cross-layer solution will provide methods of optimizing the system for size, power and reliability.

Automotive Technology: In life-critical automotive systems, advanced safety features and drive-by-wire control demand the greater computational capabilities of advanced technologies, but

also have even higher safety requirements to safeguard human life. For example, a 2008 study [45] estimated that automotive crash-avoidance systems such as blind spot detection, forward collision (following too close) mitigation, and emergency brake assist had the potential to prevent up to 3,435,000 non-fatal and 20,777 fatal crashes annually in the U.S., and a number of automakers have begun deploying limited versions of these systems in production vehicles. However, a 2009 study [113] found that 47% of the alerts produced in a road test of a more-complete collision avoidance system were so-called "nuisance" alerts that were either false alarms or alerts to situations that the driver was already aware of, arguing that more-sophisticated systems are required in order to distinguish actual threats from non-dangerous situations.

While these collision-avoidance systems have great potential, the sheer number of cars on the road places stringent demands on their reliability, particularly in the relatively-hostile automotive environment. With over 250 million cars registered in the United States, per-automobile collision avoidance system failure rates of only a few FIT could lead to multiple unnecessary accidents per day, as discussed in Section 2.2. These reliability demands are further exacerbated by the tight budget constraints of the automotive industry, which make replication-based reliability schemes impractical. Cross-layer solutions will allow designers to come up with efficient solutions that can balance size, weight, power, and reliability with performance. By sharing the responsibility for reliability across the computing stack, new technology can be used without the need to apply replication-based reliability solutions.

Airplanes: Airplane technology is life-critical for both commercial and military needs. Reliable, low-power computing systems will enable greater integration of technology in airplanes, whether to increase airplane safety, increase the comfort of passengers through entertainment systems, or increase the technology capabilities for the military.

In the case of manned air flights, the need for better technology will increase the safety of passengers and pilots. Currently, controlled flight into terrain (CFIT) is the second most common cause of approach-and-landing accidents (ALAs). In 1999, there were 7,185 fatalities from ALAs [11]. Later documents state that there have been 9,000 fatalities from CFIT and that 58% of all CFIT accidents are fatal [3]. These documents state that better technology can limit CFIT accidents. Already the increased use of Ground Proximity Warning Systems and Global Positioning Systems have decreased CFIT. In many other cases, assistive technology can make manned air travel safer, as long as they are implemented using reliable technology. Given the harsh radiation environment, adaptive systems that can seamlessly transition between the differing radiation environments will make airplane systems more reliable. Likewise, new methods in mitigating reliability problems could decrease the overhead associated with designing high-reliability systems.

6.2 Close the Technology Gap Between Aerospace and Consumer Electronics

Satellite technology provides cable television, supports in-theater warfighters, and supports science. Failures on these systems could cost millions from lost advertising revenue or loss of human life. On-board processing is necessary to optimize the limited communication bandwidth to the ground (as low as 9600 baud), but many payloads have 20-30 Watt power limits. As radiation-hardened technologies lag commercial technologies by several generations, these technologies may not meet the increasing needs of satellite systems. We must find ways to use more advanced and energy-efficient technologies without sacrificing system reliability to support the growing needs for

satellite technologies.

The effort required to design, test, and qualify electronics for aerospace applications, combined with the relatively small market for such systems when compared to consumer electronics, often causes aerospace electronics to lag 10–20 years behind the capabilities of consumer electronics. Similarly, the need to amortize design and testing costs across a much smaller volume of parts sold generally makes aerospace electronics much more expensive than their commercial counterparts. Developing cross-layer reliability will help to close this gap by allowing designers to implement systems that adapt to the reliability demands of different applications, rather than requiring complete re-implementation to meet the demands of aerospace systems.

As an illustration of the potential benefit, BAE Systems sells the RAD750 line of radiation-hardened microprocessors, which are based on the PowerPC 750 architecture. These microprocessors run at 200 MHz, and are advertised as achieving more than 400 Dhrystone MIPS [127]. An earlier, 133 MHz version of the part consumed 5W of power [32], and boards containing the RAD750 cost approximately $200,000 when the part first became available [107].

In comparison, Intel's Atom 230 processor runs at 1.6 GHz, consumes 4 Watts of power, and is available for $29 [39]. Benchmark tests of systems based on the Atom 230 achieved 4035 Dhrystone MIPS [88], giving the Atom approximately 10x higher raw performance than the RAD750, 12.5x higher performance/Watt, and 70,000x higher performance/dollar. The performance per dollar estimate is affected by the fact that the RAD750 price quoted was for a complete board, not just a processor, but indicates the magnitude of the difference between the two processors.

Developing design techniques for scalable reliability could potentially allow one product to meet the needs of both application domains. Such a processor would be configured for maximum power efficiency when running in a terrestrial environment, but would configure for maximum reliability when running in an aerospace environment. As long as the mechanisms used for maximum reliability did not impose much cost on the system when running in power-efficient mode, they could impose substantial power and performance costs when active and still have the resulting system outperform a conventional rad-hard design. Terrestrial products that used the processor could also benefit from its scalable reliability by operating in maximum-reliability mode when executing critical sections of code.

Revolutions in on-board computing will allow satellite designers to be able to provide more actionable information for the allowable bandwidth for national security missions. Data fusion on multi-sensor satellites would be possible in-situ before the data sets are returned to the ground station, which could drastically reduce the amount of data that needs to be returned for ground processing and increasing the speed that information is disseminated to policy and decision makers. Finally, the ability to distribute quality data in real-time to support the warfighter can be provide a tactical advantage that can help protect soldiers during battle.

6.3 Save Energy and Improve Service Through Smart Infrastructure

With recent oil crises, the need for energy security, autonomy, and reduction is a high priority. One way to achieve these goals is through Smart Infrastructure [66]. In [66] it is stated that it would take "hundreds of billions of dollars in conventional electric infrastructure over the next 20 years to meet expected load growth." The GridWise system proposed in [66] would provide "higher asset utilization" and "increased efficiency" by using information technology to provide more control and interaction with the infrastructure.

With an increased use of computing technology to provide a more efficient and responsive system for the infrastructure grids, reliability and security are a high priority. The Northeast Blackout of 2003 left 55 million people without power, and contributed to eleven fatalities [2]. Anderson Economic Group put the cost of lost earnings from the blackout at $6.4 Billion USD. Therefore, as the infrastructure embraces more technology to provide increased system services, reliability will be necessary. Full-scale reliability is difficult with such a large system, because the entire system spans many different locations, community sizes, and system types. Cross-layer reliability techniques will be useful so that reliability can be tuned to meet the needs of the particular system and the system location. In that manner, a large-scale, high-mountain system, such as would be used to support Denver, CO, could adapt to a potentially higher error rate than a smaller system supporting a rural farming community. Furthermore, legacy systems could be supported through adaptations of the higher levels of the computing stack, such as middleware, operating systems, and applications (See Section 3.4).

6.4 Increase Security on National Mission Computing Platforms

As illustrated by a number of recent attacks that exploit transient hardware errors to compromise a system's security mechanisms [134, 116, 94, 54], a computer system cannot be secure unless its data and computations are reliable. Cross-layer reliability techniques will help future computer systems provide the security required for electronic commerce, electronically-stored medical records, and military applications by providing a solid, reliable, foundation on which software security mechanisms can be built.

6.5 Decrease Human Liability for In-theater Warfighters

Technological advantage has always been an important part of supporting our soldiers during war. Recent innovations for several national security missions have used unmanned aerial vehicles to provide warfighter support. Persistence surveillance of battlefields can provide the infantry an important global view of potentially dangerous scenarios when delivered to them in handheld, real-time systems. Systems that can quickly and accurately determine the locations of land minds and improvised explosive devices can protect soldiers navigating foreign territories. These systems are likely to use a heterogenous computing environment that will rely on satellite systems, UAVs, ruggedized standard computers, and handheld devices and could be composed of hundreds to thousands of different computing components.

Unreliable computing systems can cause loss of human life in these scenarios. Furthermore, large, heterogenous computing environments that span operating environments of terrestrial and space computing, can be difficult to develop, as different systems will have different reliability needs based on the mission and operating environment. On top of it, the entire computing environment will need to adapt as a rapidly evolving battlefield environment might include the permanent or temporary loss of computing elements, including those elements providing information about battle environment. Given the scale of the systems and the need for high-reliability applications, cross-layer reliability techniques will allow the reliability features to adapt to the system location, mission requiremnets and system needs.

During the two current wars, there has been an increase in the use of unmanned air flights. For these systems the need for high-quality, high-reliability technology solutions is very much

needed. As UAVs become smaller and cheaper, there will be a need to use lighter-weight reliability methods that can increase the reliability of the unmitigated system without sacrificing size, power and weight. UAVs could also play an important role for in-theater warfighter support by either taking the place of on-the-ground soldiers or by giving soldiers a reliable overhead view of the battlefield. In both uses, technological solutions will help prevent loss of life during battle.

7 Education

Changing Electrical Engineering, Computer Engineering, and Computer Science curricula[2] are essential to responsible continued exploitation of computer-mediated systems throughout our modern society.

Reliability is an important part of almost all engineering disciplines since the failure of engineered artifacts affect human lives. Civil engineers design buildings, bridges, and dams; Mechanical engineers design cars, motors, engines; Nuclear engineers design power plants and nuclear material handling; Chemical engineers design plants and processes that contain and mix dangerous chemicals; Aerospace engineers design airplanes and rockets; Computer engineers design the systems that control cars, airplanes, plants, medical devices, and communication and financial systems. *However, reliability is not currently a core requirement in computer engineering related degrees.* This is largely an effect of education not keeping up with the rate of change in our world. Our high modern reliance on networks and servers for communication and electronic and computer-mediated control of critical infrastructure (power grids, transportation, financial) is a transition we have made over the past two decades. Embedding of computerized control in safety-critical systems (drive-by-wire, medical prostheses) is also something that has recently become feasible with the last 20 years of Moore's law scaling. As a result, today's electrical engineering curricula only deal with reliability systematically in communication systems (that date back five decades). Similarly, computer science curricula evolved around mainframe, then desktop, and now web computing where failures are inconvenient, but seldom life critical. In both programs, reliability hardware and software shows up, at best, as a specialty elective rather than a core requirement or a pervasive cross-cutting theme.

The success of computer-mediated control creates new risks that, in some ways, make them more dangerous than physics-based engineered structures. Information-dominated design costs, enabled by inexpensive electronics, makes it possible to create and deploy computerized designs rapidly with small capital costs. The design and deployment of a bridge, airplane, or chemical plant demands considerable cost for raw materials and time for deployment. As a result, it is impossible to build and deploy a system without scrutiny at multiple levels, and the cost of assessing safety and reliability up front is small compared to the cost of materials and labor for construction. In contrast, with cheap electronics and general-purpose programmable components (microprocessors, FPGAs), most of the cost lies in the design of the software. This makes it possible to build solutions without large capital outlays. It also allows a solution, once designed, to be rapidly and inexpensively deployed. These are the great strengths of computerized-control and part of why it is rapidly being deployed into all engineered systems. However, it also means that solutions can be developed with less oversight, can be developed by individuals with no reliability concern, and the cost of proper reliable design may actually be the dominant cost in development. Systems not originally developed for critical roles can be rapidly into critical roles, often with little review.

Given the role of computer-mediated control in today's society, the trend and economic benefits of using general-purpose system platforms, the increasing vulnerability of hardware, and this ability to rapidly redeploy **any** *design into a critical role, it is reasonable to demand as much safety and reliability education of all computer engineers as we do of aerospace, civil, chemical, and nuclear engineers.* This way all computer engineers are sensitized to the issues, know how to approach safety design and assessment, and know when this must be considered. Since the ACM

[2]Hereafter referred to collectively as computer engineering.

and IEEE professional societies, through their joint curriculum committees, provide the leadership for recommending curricula and accreditation bodies, such as ABET, defer to the professional societies for curriculum guidance, it will be important to encourage these changes in their curricula recommendations.

One of the key challenges to implementing these revisions, and even to getting agreement to recommend these changes, is being able to point to (1) an appropriate body of science and engineering work to build upon, (2) the textbooks and pedagogical materials to support education, (3) projects and platforms suitable for instruction, and (4) pool of experts in academia prepared to guide instruction. A common complaint from faculty for core treatment of reliability on the software side is the lack of an adequate body of scientific work to build upon. This study further recommends a number of key areas where we need to develop greater scientific and engineering understanding with the goal of expanding the knowledge available (Section 5). Therefore, beyond the research recommended here, we also recommend supporting:

- Development of pedagogical materials that can be widely used in the academic community (text books, web-distributed lecture notes and knowledge bases)
- Workshops to develop specific curricula recommendations
- Common platforms for study, analysis, and experimentation (further developed in Section 10)
- Workshops for refreshment (life-long learning) of existing computer-engineering faculty
- Prioritized training support (fellowships, training grants) for graduate students heading to academic careers with strong connection to safety and reliability of computer systems

8 Critical Questions

Beyond the direct technical questions for the research, a number of logistical, sociological, and broader technical questions arise as potential areas of risk. These include:

- How do we enable concurrent research across the stack?
- How do we avoid making system fault rate worse by increasing the complexity of software and, as a consequence, the rate of software failures?
- How do we avoid increasing the burden on the programmer?
- How do we support legacy application?
- Should security issues be addressed as an integral part of this research?
- How should passives and analog device reliability be addressed?
- How do we avoid increasing risks through backup complacency?

8.1 Concurrent Research

Cross-layer design means that changes need to occur at multiple layers. We cannot sequence the research resolving one layer at a time. The demand for changing interfaces is both an opportunity and a key logistical challenge. This necessitates a period of "chaos" in research as we explore useful information and how interfaces might change. Section 10 touches further on this issue as it is one of the key challenges to organizing effective research programs.

8.2 Software and Applications

The next three questions all address issues with software. We first elaborate the concerns, then discuss how these concerns should impact research and solutions.

8.2.1 Concerns

Software Complexity Software faults are responsible for a significant fraction of system outages. This arises in part because modern software complexity often exceeds the capabilities of our the available software engineering technique. In our cross-layer approach, we potentially ask software to play a larger role in systems. This increases the software that can go wrong, increases the interactions that may occur with software, and means software may be able to make fewer assumptions about the correctness of the hardware.

Programmer Burden Application participation in hardware error mitigation may place more demands on already overloaded programmers. With human time at a premium, this can increase costs and time-to-market.

Legacy Existing applications were not built to the new interfaces. It will not be feasible to demand every existing application be rewritten to new interfaces.

8.2.2 Impact

Layers As noted in Section 11, cross-level mitigation does not necessarily mean that the application programmer bears the burden for reliability. Many layers exist between the hardware and the application that can contribute. Some of these require software work (*e.g.* compilers, operating systems, middle ware), but that work falls to experts at these intermediate layers and is then available to all applications, including legacy applications.

Nonetheless, assistance from the application can allow greater efficiency than solutions that have no application assistance. Good tools for identifying the weak links in applications and the bottlenecks in overhead can help focus programmer efforts on the places for highest impact. As it is typical today to write programs in a simple way, identify the performance bottlenecks, and then optimize only those portions of the program that contribute to the performance bottleneck, in the future, we can provided automated support below the application programmer level and feedback to the programmer so she can understand where the lower-level mitigation support is most expensive so she can focus her effort on providing the information that will most assist the lower layers in finding an efficient solution. This underscores the need for "Design Guidance" metrics (Section 9.1).

Raise Level of Abstraction Many of the cross-layer solutions should raise the abstraction level at which the programmer operates. As such, these techniques should also take away some of the programmers concerns (*e.g.* selection of encodings and particular performance and protection techniques) resulting in a net reduction in programmer burden. The programmer is asked to provide high-level goals and invariants, allowing the co-operative multi-layer optimization to assess and select specific implementations.

Software Engineering A more ambitious perspective views the individual hardware and software reliability solutions as part of an integrated solution for the full system reliability problem. A natural question that arises is: can we leverage the ongoing research to address software reliability for the purpose of hardware reliability and vice versa? That is, should we be solving these problems together? This is a rich, and mostly unexplored, area of investigation with significant opportunities for improving the overall cost and quality of resilience.

Because of the aforementioned challenges with software faults, there is a significant movement in industry today to adopt software practices that improve software resiliency. These cover the gamut of detection, diagnosis, and recovery techniques for software bugs through methods such as formal assertion frameworks and interface specifications for bug detection, transactional semantics for recovery, and rollback/replay based production time diagnosis. If we take the view that the only hardware faults that matter are those that affect software behavior, then we can treat a hardware fault effectively like a software bug. In theory, we can apply many of the software fault detection and recovery techniques for hardware faults, amortizing the costs of those techniques across both hardware and software reliability. Conversely, hardware support can be amortized for both hardware and software reliability. Recent research has provided evidence for the potential of such an approach [111] but barely scratches the surface. We still do not fully understand how the plethora of possible hardware faults manifest in software, and what type of software level detection, diagnosis, and recovery techniques would work for such manifestations.

A potential argument against such a research agenda is that software resiliency techniques are still evolving and it is premature to try to leverage them. However, this period of evolution is exactly the right time to invest in this area—once software resiliency practices become mature, it may become too late to adapt them for and to exploit hardware resilience.

8.3 Trusted Computing

During the course of the workshops and the writing of the final report, the idea of combining reliability and security problems gained a significant amount of traction at mission agencies. While it is understandable that the mission agencies are concerned about both problems and would prefer a singular solution, it is not clear there is a single "silver bullet" that addresses both problems completely. As noted, reliability is *necessary* to achieve security (Section 6.4). However, reliability solutions alone are not sufficient to address the security challenges. Part of the problem stems from the differing attack models. The fault models for reliability cover many different types of failures from yield failures to transient radiation failures. In some cases, mitigation methods can be used to increase the robustness of a system to multiple fault modes, but there are also times when mitigating one fault model can cause reliability problems from another fault model to increase (See, for example, the transient versus persistent fault metric discussion in Section 9). Security threat models fundamentally differ from fault models. While reliability problems predominantly manifest as random faults, threat models are adversarial in nature and may be highly non-random. Nonetheless, many of the layer interfaces, information sharing, error filtering, availability enhancement, and repair techniques may be useful for security as well, creating opportunities to unify portions of the security and reliability solutions. However, some of these interfaces could create new opportunities for attacks and disclosure of information valuable to an adversary. Furthermore, the adversarial nature of security threats means security will require either additional techniques to detect and prevent adversarial attacks or to guarantee that adversarial attacks can never be stronger than random attacks. These additional techniques and considerations were beyond the scope of this study and will merit additional attention.

8.4 Analog Devices

During the course of the workshops, the reliability of analog devices was raised repeatedly. In many cases, analog devices are used to convert between analog and digital signals. For these cases the reliability of the analog devices directly affects the reliability of input and output data from the system. For medically implantable devices, reliable input data is necessary to determine how the device interacts with the body. For automotive systems, reliable input data is necessary to determine how to react to a rapidly changing environment while the car is in motion. For satellite systems, reliable input data is neccessary for triggering event monitors or data collection needs. For many of these systems, the least reliable components in the system are the analog devices devoted to converting between analog and digital signals.

In many ways, current reliability standards and methodologies focus on reliable computation and assume that the input or output information for system is reliable. This assumption might not be as true as it was in the past. While reconfiguration to adapt to analog variation and improvements in light-weight filtering methods or error filtering could help improve the reliability of the information stream, the science questions posed in Section 5 do not address analog reliability issues as directly as they do computational reliability.

Finally, in the satellite environment, efficient DC-to-DC converters are necessary for using new hardware devices with lower voltages, but most of these devices lack the necessary radiation hardness for multiyear missions. High quality, radiation-hardened DC-to-DC converters will be necessary for satellite designers to adopt new hardware, especially commercial processing electronics. Therefore, any attempts to "mainstream" satellite technology onto commercial technology

roadmaps will need to include research and development into DC-to-DC converters.

While it is clear that analog component reliability must also be addressed, aside from them interactions noted (Table 2), it is not clear the solutions needed for analog overlap with the solutions needed for computational reliability. As such, these issues likely need to be addressed in a distinct research program.

8.5 Backup Complacency

When we examine major disasters (*e.g.* Three Mile Island, BP Oil Spill in Gulf of Mexico), we often find that operators take inappropriate risks assuming the safety interlocks will prevent catastrophic failures. We also see that the safety interlocks are often poorly maintained such that they are not available when needed. In fact, these disasters occur and we know about them precisely because the layered backup systems fail as well (when the layered backups succeed, disasters are averted, the public seldom learns about them, and investigations are either not performed or not openly published). Similarly, there is some evidence that people drive less safely knowing that anti-lock brakes will compensate for their shortcomings and seat belts and airbags will reduce the damage they may suffer in an accident.

Part of the system knowing its own health (Section 5.7) is the continual self-diagnostic of available capacity and reliability provided. This is one of the benefits of automating the reliability assessment and making it part of the runtime operation of the systems (Section 9.1). That is, we avoid the problem of safety margins and interlocks not being there when we need them by continually assessing the system protection as an integral part of operation. Current practice where reliability computations are not automated, not composable, and depend on assumptions of part reliability are more suspect to the backup complacency problem than the vision advocated here. If operators are allowed to control the reliability target, they can still err by setting it too low for the mission. Providing good user interfaces and simple and understandable models for the operator will always be important and will be essential to the successful deployment of these techniques.

9 Metrics

During the study, there was frequent discussion of the challenges posed by the lack of good metrics for reliable system design. Current reliability metrics, such as failures-in-time, mean-time-between-failures, and silent data corruption rates do not provide good guidance for design and optimization. These metrics only measure the quality of a design, and do not capture the power, performance, and area costs of achieving that quality. Because of this, designers cannot conclude that a change that improves the value of a metric is an improvement to the design. In addition, many of these metrics do not *compose* well, meaning that it is difficult or impossible to predict how a change in the value a sub-system achieves on a given metric will affect the overall system's score on that same metric. This makes existing metrics difficult to apply in the early stages of the design process, which is typically when a design is most flexible.

Finally, current reliability metrics are too dependent on the details of the fabrication process used to implement a system. In particular, most existing metrics cannot be computed on a system without knowing the fault, variation, and aging rates that the system will experience, which are strongly dependent on the fabrication process. This makes it difficult to use these metrics for research or design, because it is difficult to accurately predict the fault, variation, and aging rates of future fabrication processes. In addition, many companies consider these quantities to be highly proprietary data, making it difficult to obtain accurate values even for existing fabrication processes.

A research program in cross-layer reliability should include research into reliability metrics. Better metrics will allow researchers to evaluate their designs and to be confident that their techniques address the right problems. More importantly, better metrics will enable work on automatic optimization of reliable systems by providing good objective functions that optimizers can target. Good metrics will also be useful in managing a research program because they will allow program managers to compare and evaluate researchers' results.

In this section, we present the study's conclusions about reliability metrics. We begin by outlining the characteristics that good reliability metrics should have in order to set goals for metrics research. We then suggest some promising starting points for reliability metric research.

9.1 Characteristics of Good Metrics for Reliability

Metrics for cross-layer reliability should:

Enable Optimization We need metrics that are comprehensive enough that designers can agree that a design that meets the target values of the reliability metric(s) while also satisfying the other design requirements (power, performance, etc.) is an acceptable solution. Comprehensive reliability metrics will also reduce design effort by making it possible for designers to trust the results of automated optimization techniques. Good metrics for optimization will also be critical to the development of self-monitoring systems, as they will allow the system to determine when it needs to adapt to changing conditions or when it can no longer meet its design specifications.

Many current metrics are *incomplete*, and only capture some of the vulnerabilities of a design. Using incomplete metrics to evaluate a design can lead to bad decisions when choices that improve the metric make the design worse in some other important dimension. For example, optimizations to reduce architectural vulnerability factors [83] can decrease the rate of propagation of transients to the architectural level; however, they can also increase the chance of accumulating an undetectable collection of hard errors [41]. Comprehensive metrics that adequately bound all effects will be

necessary to enable broader use of optimization. Metrics that capture efficiency as well as reliability (*e.g.*, some reliability analog to the energy-delay product of a processor or ASIC) would also be extremely useful (See Section 5.6.2).

Be Predictive and Composable We further need metrics that are adequately predictive and allow composition and cross-layer mediation. We need predictive metrics to allow designers to estimate system reliability before the design is complete, while it is still possible to make changes. Currently, the most accurate reliability metrics require observation of the complete system, either in its intended operating environment or in one that yields increased failure rates such as a radiation beam. This is both too expensive and happens too late, after the design has been completed. Furthermore, full-system observation is not a reasonable metric to embed in an automated optimization loop.

Similarly, we need metrics that are composable, so that designers can estimate how design decisions will affect overall system reliability. As an illustration, Amdahl's Law allows the composition of performance metrics of different portions of a system by noting that a technique's impact on overall performance is proportional to the fraction of execution time where the technique is useful. Some reliability equivalent to Amdahl's Law would greatly facilitate reliable system design.

Composition metrics that simply add the failure rates of components are inadequate, because they do not account for the structure of a design or the impact of mitigation techniques (*e.g.* simply adding the failure rates of a system with replicated components would give the impression that adding redundancy lowers reliability, which is, hopefully, not the case). In addition, systems are not equally sensitive to all components of their design (Section 5.5), and individual components will not have equal sensitivity to error rates.

Provide Guidance to Designers Metrics that estimate the overall reliability of a system may not provide the information required during the design process, particularly at higher levels of the system. There is a strong need for metrics that help guide the design process in addition to computing system reliability, and this need is expected to grow as inter-layer cooperation and automated optimization become more common.

To support inter-layer cooperation, metrics should identify both the elements that are most likely to cause system failures and the types of errors that are most likely to escape a given element or stack level's reliability mechanisms. This will help designers understand where more information and control need to pass between components (Section 5.3), how the filtering mechanisms (Section 5.2) at higher levels of the stack interact with the errors that manifest in lower levels, and where additional mechanisms can most effectively be added to improve system reliability.

Designer guidance metrics as feedback from automatic optimizations are also needed to identify the weak links in a system's reliability scheme and provide insight into the nature of these weaknesses. This information will help designers decide where automatic optimization alone will be able to meet reliability goals, where more alternatives and tuning options would help optimizers improve reliability, and where human intervention is necessary. For example, a good design guidance metric might identify cases where adding additional invariants to a design would make the design more amenable to automatic optimization, identify areas where application-specific checks could significantly reduce hardware reliability overhead, or otherwise assist the designer in identifying those areas where a small amount of human effort would have a large impact on system quality.

9.2 Starting Points for Metrics Research

While metrics for reliability are an open problem, and have been for some time, the study group was able to identify four promising areas for research. Progress in any one of these areas would significantly contribute to achieving the goals described above, and we outline them here to suggest potential starting points for metrics research:

Fabrication-independent metrics for noise rates As discussed above, it is difficult to obtain accurate estimates of the rates of faults, variation, and aging in future fabrication processes, due both to the proprietary nature of reliability data for current processes and the uncertainties involved in predicting quantities that researchers are actively trying to improve. In the worst case, predictions of future noise rates become self-invalidating, because fabrication researchers use those same predictions to guide their efforts.

There is a strong need for technology-independent metrics that characterize the noise rates seen by electronic systems. Such metrics will necessarily be a combination of defects, variation, wear rates, and transient upset rates and will further require characterization of the distribution of effects (*e.g.* random, clustered, adversarial). Similarly, noise rate characterizations should also capture the sensitivity of noise rates to operating environment (altitude, temperature, etc.).

Metrics that combine costs, reliability, and performance System design is a multi-dimensional optimization process, requiring trade-offs that improve quality in some dimension at costs in others. We need metrics that capture the most critical dimensions of a design (power, performance, and reliability) in order to assist designers in finding optimal design points. A potentially valuable multi-dimensional metric might quantify the energy a system requires to perform an operation at a given performance target (quality of service (QoS), operation latency, etc.) and noise rate. Alternately, a metric might express energy/operation as a function of performance and noise (E(QoS, noise)).

This metric would reflect the fact that energy has replaced chip area as the dominant efficiency concern in electronic systems. It would also reflect the fact that perfect reliability is an impossible goal, making it necessary to define acceptable rates and severity of user-visible errors. Most importantly, because it captures the interaction between the three most-important dimensions of design quality, E(QoS, noise) is a good metric for manual or automatic optimization, since designers can be confident that changes that improve the value of the metric will improve the overall quality of the design.

Metrics that consider rates and severity of failures In order to understand whether a system is reliable enough for a given application, it is important to understand both how often faults become visible to the end user and how severe the impact of those faults are. For example, when discussing faults that make a system unavailable for some time, it is important to consider both the rate of occurrence and the length of outage [122]. An anti-lock braking system with an expected downtime of 10 seconds/year might be acceptable if that downtime occurred as 1000 10msec outages per year, but would likely cause fatal crashes if the downtime occurred as one 10-second outage each year. Having metrics that capture both fault rate and severity would significantly assist designers.

Metrics for adaptation As systems become capable of adapting to changes in their environment or internal state, it will be necessary to have metrics to evaluate the quality and costs of this adaptation. Metrics will be needed to evaluate how well a system adapts to noise rates outside of the operating range for which it is currently optimized, how quickly a system adapts to failures, and how much energy it expends in adaptation. In particular, it will be important to be able to characterize the quality of a system's adaptation (how its best configuration for a situation compares to an

ideal design for that situation) and the costs of its mechanisms for adaptation (*e.g.*, is a design that is always within 10% of optimal but has power-hungry adaptation hardware better or worse than one that stays within 20% of optimal, but has simple adaptation hardware?).

10 Research Organization and Infrastructure

The designers of a research program in cross-layer reliability will need to address two key technical and organizational challenges in order for the program to be effective while remaining within the budget and time constraints of most funding agencies.

On the technical side, a lack of widely-accepted models for failure rates, variation, aging, and fault manifestation (Section 9.2) makes it difficult for researchers to do quantitative analysis without measured data from fabricated circuits. This data is often difficult or impossible to obtain, either because the fabrication processes being studied do not exist yet, because the processes do exist but are new enough that measured data is highly-proprietary information, or because of the cost of fabricating and testing prototype chips is prohibitive. When it is possible to obtain empirical data, doing so often takes long enough that the data has limited predictive value in the absence of good models. For example, the designed lifetime of many integrated circuits is on the order of 5-7 years, while new fabrication processes are typically introduced every 2-3 years. By the time a full-lifetime aging study of chips fabricated in a given process completed, that fabrication process would be at least one, and possibly as many as three, generations behind the state-of-the-art, making it less useful to researchers without models to use that data to predict aging in future fabrication processes. (Nonetheless, such data would be extremely useful to designers of life-critical systems, who often choose to use older fabrication processes specifically because the aging behavior of those processes is well-known)

Research programs in cross-layer reliability will also be challenged by the lack of existing, "standard" models of cross-layer reliable systems. This is a traditional problem in any new area of research, where the lack of previous work forces researchers to develop entire systems "from scratch." As the field becomes more mature, it becomes possible for researchers to assume that most of a system is constructed according to a conventional model and focus on the particular piece of the system that interests them. For example, computer architecture is now a mature enough field that researchers studying memory system design can assume, and obtain models of, conventional microprocessor architectures and change only the memory hierarchy, which both reduces the amount of effort required to investigate an idea and allows more-even comparisons between different approaches to the same problem.

Because of this, an initial research program in cross-layer reliability will need to encourage tight collaborations and rapid exchange of results between researchers working at all levels in the system stack. To prevent reliability from delaying the rate of progress in silicon fabrication, researchers will need to address all levels of the system stack in parallel rather than serializing on a top-down or bottom-up approach, but will also need to quickly incorporate the results of other groups' work. In the past, programs have provided some of this tight collaboration by funding large centers, but funding a small number of centers at this point would limit the number of new ideas explored at this early stage.

To address these concerns, we recommend that a research program in cross-layer reliability:

- Have as a goal the development of one or more open frameworks/models for cross-layer reliability that future work can modify or add to. It should be clear that the model composition and interfaces is part of the research so will, itself, be a subject of study, experimentation, and evolution during the research.

- Support the development of tools, metrics, models, and benchmarks that

  - Characterize how device-level physical effects propagate to system-visible errors
  - Support performance/accuracy trade-offs and/or multi-level modeling to allow both analysis of complete systems and detailed simulation of low-level effects
  - Enable analysis of fault-tolerant systems without requiring accurate fault rates as an input, for example by generating curves that show system reliability as a function of fault rate
  - Predict how fault rates will change as fabrication technology advances

- Support and encourage tight collaboration between teams without requiring that researchers be located at the same facility. At a minimum, the program should support regular meetings/symposia and encourage exchange of tools and simulations. Another possibility would be to reserve some of the program's funding to support collaborations that develop over the course of the program, encouraging researchers to work together

11 Layers and Communities

Throughout this document, we have discussed layering and co-operation across the layers. Following is a non-exhaustive identification of typical layers in modern systems:

- application
  - code developed by application programmer
  - design patterns and software architectures
  - languages (potentially multiple languages with some specialized to the application domain)
  - compilers
  - libraries
- protocols and services
- middleware
- platform
- operating system
- virtual machine
- architecture
- microarchitecture
- circuits
- devices

This expanded list helps us see the scope of intellectual disciplines that will be touched by this research. It also points out that there are many layers between circuits and the programmer where co-operative mitigation may occur. Consequently, there are many places (*e.g.* middleware, libraries, compilers) where software can apply co-operative mitigation techniques without burdening the application programmer.

In addition to the research communities naturally associated with these layers, many other communities can and should be engaged in this endeavor, including:

- co-design
- computer-aided design
- control theory
- information theory and communications
- learning and artificial intelligence
- signal processing
- software engineering
- theory and algorithms

Many areas and industries will potentially benefit from these developments. We have engaged many of these communities in our constituency groups (Section 4) and expect they will continue to be important participants in the research, including:

- aerospace
- cloud computing
- consumer electronics
- data centers

- electrical grid
- embedded real-time and CyberPhysical systems
- financial services
- industrial control
- medical instruments
- military
- networking
- robotics
- supercomputing
- transportation

References

[1] Intelsat rogue satellite: Nothing to worry about. on web at http://www.networkworld.com/community/node/61291 last accessed on May 17, 2010.

[2] Northeast blackout of 2003. on web at http://en.wikipedia.org/wiki/Northeast_Blackout_of_2003, last accessed 7/6/2010.

[3] Prevention of controlled flight into terrain in general aviation operations. on Web at http://www.americanflyers.net/aviationlibrary/CFIT.asp, last accessed July 1, 2010.

[4] U.S. to shoot down satellite wednesday, official says. on web at http://www.cnn.com/2008/TECH/space/02/19/satellite.shootdown/index.html last accessed on May 17, 2010.

[5] Pacemakers. on web at http://www.chfpatients.com/implants/pacemakers.htm, last accessed on July 1, 2010, 2007.

[6] International technology roadmap for semiconductors. <http://www.itrs.net/Links/2008ITRS/Home2008.htm>, 2008.

[7] International technology roadmap for semiconductors. <http://www.itrs.net/Links/2009ITRS/Home2009.htm>, 2009.

[8] U.S. Defense Advanced Research Projects Agency. DARPA grand challenge and DARPA urban challenge. on web at http://www.darpa.mil/grandchallenge/index.asp, 2007.

[9] Nidhi Aggarwal, Parthasarathy Ranganathan, Norman P. Jouppi, and James E. Smith. Configurable isolation: building high availability systems with commodity multi-core processors. *SIGARCH Comput. Archit. News (ISCA 2007)*, 35(2):470–481, 2007.

[10] Greg Allen, Gary Swift, and Carl Carmichael. Virtex-4VQ static SEU characterization summary. Technical Report 1, Xilinx Radiation Test Consortium, 2008.

[11] Flight Safety Foundation Approach and-landing Accident Reduction Task Force. Analysis of critical factors during approach and landing in accidents and normal flight. *Flight Safety Digest*, pages 1–256, 1999.

[12] J.M. Armani, G. Simon, and P. Poirot. Low-energy neutron sensitivity of recent generation SRAMs. *Nuclear Science, IEEE Transactions on*, 51(5):2811–2816, Oct. 2004.

[13] Todd Austin, David Blaauw, Trevor Mudge, and Krisztain Flautner. Making typical silicon matter with Razor. *Computer*, 37(3):57–65, 2004.

[14] Todd Austin, David Blaauw, Trevor Mudge, and Krisztián Flautner. Making typical silicon matter with Razor. *IEEE Computer*, 37(3):57–65, March 2004.

[15] Kevin J. Barker, Kei Davis, Adolfy Hoisie, Darren J. Kerbyson, Mike Lang, Scott Pakin, and Jose C. Sancho. Entering the petaflop era: the architecture and performance of roadrunner. In *Proceedings ACM International Conference on Supercomputing*, pages 1–11, 2008.

[16] H.J. Barnaby, C.R. Cirba, R.D. Schrimpf, D.M. Fleetwood, R.L. Pease, M.R. Shaneyfelt, T. Turflinger, J.F. Krieg, and M.C. Maher. Origins of total-dose response variability in linear bipolar microcircuits. *Nuclear Science, IEEE Transactions on*, 47(6):2342–2349, Dec 2000.

[17] H.J. Barnaby, R.D. Schrimpf, A.L. Sternberg, V. Berthe, C.R. Cirba, and R.L. Pease. Proton radiation response mechanisms in bipolar analog circuits. *Nuclear Science, IEEE Transactions on*, 48(6):2074–2080, Dec 2001.

[18] Luiz André Barroso and Urs Hölzle. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*. Number 6 in Synthesis Lectures on Computer Architecture. Morgan & Claypool, 2009.

[19] Joel Bartlett, Wend Bartlett, Richard Carr, Dave Garcia, Jim Gray, Robert Horst, Robert Jardine, Doug Jewett, Dann Lenoski, and Dix McGuire. *Reliable Computer Systems: Design and Evaluation*, chapter The Tandem Case: Fault Tolerance in Tandem Computer Systems. A K Peters, Ltd., 1998.

[20] M. Berg, C. Poivey, D. Petrick, D. Espinosa, A. Lesea, K.A. LaBel, M. Friendlich, H. Kim, and A. Phan. Effectiveness of internal versus external SEU scrubbing mitigation strategies in a Xilinx FPGA: Design, test, and analysis. *Nuclear Science, IEEE Transactions on*, 55(4):2259–2266, Aug. 2008.

[21] David Birnie, Kathryn Williams, Ann Guoa, Lisa Mielniczuk, Darryl Davis, Robert Lemery, Martin Green, Michael Gollob, and Anthony Tang. Reasons for escalating pacemaker implants. *The American Journal of Cardiology*, 96(1):93–97, 2006.

[22] Jason Blome, Shuguang Feng, Shantanu Gupta, and Scott Mahlke. Self-calibrating online wearout detection. In *Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 109–122. IEEE Computer Society, 2007.

[23] Manuel Blum and Sampath Kannan. Designing programs that check their work. *Journal of the ACM*, 42(1):269–291, January 1995.

[24] Colin Blundell, Joe Devietti, E. Christopher Lewis, and Milo M. K. Martin. Making the fast case common and the uncommon case simple in unbounded transactional memory. In *Proceedings of the International Symposium on Computer Architecture*, pages 24–34, 2007.

[25] Jonathan Bohren, Tully Foote, Jim Keller, Alex Kushleyev, Daniel Lee, Alex Stewart, Paul Vernaza, Jason Derenick, John Spletzer, and Brian Satterfield. Little ben: The ben franklin racing team's entry in the 2007 DARPA urban challenge. *Journal of Field Robotics*, 25(9):598–614, 2008.

[26] David Bol, Renaud Ambroise, Denis Flandre, and Jean-Didier Legat. Interests and limitations of technology scaling for subthreshold logic. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 17(10):1508–1519, 2009.

[27] Shekhar Borkar. Designing reliable systems from unreliable components: the challenges of transistor variability and degradation. *IEEE Micro*, 25(6):10–16, November–December 2005.

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

[29] Fred A. Bower, Daniel J. Sorin, and Sule Ozev. A mechanism for online diagnosis of hard faults in microprocessors. In *Proceedings of the 38th annual IEEE/ACM International Symposium on Microarchitecture*, pages 197–208, Barcelona, Spain, 2005. IEEE Computer Society.

[30] K. A. Bowman, J. W. Tschanz, Nam Sung Kim, J. C. Lee, C. B. Wilkerson, S.-L. L. Lu, T. Karnik, and V. K De. Energy-efficient and metastability-immune resilient circuits for dynamic variation tolerance. *IEEE Journal of Solid-State Circuits*, 44(1):49–63, January 2009.

[31] Keith Bowman, James Tschanz, Chris Wilkerson, Shih-Lien Lu, Tanay Karnik, Vivek De, and Shekhar Borkar. Circuit techniques for dynamic variation tolerance. In *Proceedings of the 46th Annual Design Automation Conference*, pages 4–7, San Francisco, California, 2009. ACM.

[32] Laura Burcin. RAD750 experience: The challenges of SEE hardening a high performance commercial processor. Slides from 2002 Microelectronics Reliability and Qualification Workshop, `http://www.aero.org/conferences/mrqw/2002-papers/A_Burcin.pdf`, 2002.

[33] M. Caffrey, W. Howes, D. Roussel-Dupre, S. Robinson, A. Nelson, A. Salazar, M. Wirthlin, and D. Richins. On-orbit flight results from the reconfigurable Cibola flight experiment satellite (CFESat). In *Field-Programmable Custom Computing Machines 2009*, 2009.

[34] Michael Caffrey, Keith Morgan, Diane Roussel-Dupre, Scott Robinson, Anthony Nelson, Anthony Salazar, Michael Wirthlin, William Howes, and Daniel Richins. On-orbit flight results from the reconfigurable cibola flight experiment satellite (CFESat). In *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines*, pages 3–10, 2009.

[35] G. Candea, S. Kawamoto, Y. Fujiki, G. Friedman, and A. Fox. Microreboot–a technique for cheap recovery. In *Proceedings of the 6th Symposium on Operating Systems Design and Implementation*, pages 31–44, 2004.

[36] Nicholas P. Carter, Helia Naeimi, and Donald S. Gardner. Design techniques for cross-layer resilience. In *Design and Test in Europe (DATE)*, 2010.

[37] L. Condra, J. Qin, and J.B. Bernstein. State of the art semiconductor devices in future aerospace systems. In *Proceedings of the FAA/NASA/DoD Joint Council on Aging Aircraft Conference*, April 2007.

[38] Kypros Constantinides, Onur Mutlu, Todd Austin, and Valeria Bertacco. Software-based online detection of hardware defects: Mechanisms, architectural support, and evaluation. In *Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 97–108. IEEE Computer Society, 2007.

[39] Intel Corporation. Intel atom processor family. `http://ark.intel.com/ProductCollection.aspx?familyID=29035`.

[40] J.-P. David, F. Bezerra, E. Lorfvre, T. Nuns, and C. Inguimbert. Light particle-induced single event degradation in SDRAMs. *Nuclear Science, IEEE Transactions on*, 53(6):3544–3549, Dec. 2006.

[41] André DeHon. The case for reconfigurable components with logic scrubbing: Regular hygiene keeps logic fit (low). In *Proceedings of the International Workshop on Design and Test of Nano Devices, Circuits, and Systems*, pages 67–70, September 2008.

[42] André DeHon and Helia Naeimi. Seven strategies for tolerating highly defective fabrication. *IEEE Design and Test of Computers*, 22(4):306–315, July–August 2005.

[43] Anand Dixit, Raymond Heald, and Alan Wood. The impact of new technology on soft error rates. In *Proceedings of the IEEE Workshop on Silicon Errors in Logic – Systems Effects*, 2010. Available online: `http://www.selse.org/Papers/14_Dixit_P.pdf`.

[44] Gasiot et al. Multiple cell upsets as the key contribution to the total SER of 65 nm CMOS SRAMs and its dependence on well engineering. *IEEE Transactions on Nuclear Science*, 54(6):2468–2473, Dec 2007.

[45] Charles M. Farmer. Crash avoidance potential of five vehicle technologies. Insurance Institute For Highway Safety White Paper, 2008.

[46] David Fick, Andrew DeOrio, Jin Hu, Valeria Bertacco, David Blaauw, and Dennis Sylvester. Vicis: a reliable network for unreliable silicon. In *Proceedings of the 46th Annual Design Automation Conference*, pages 812–817, San Francisco, California, 2009. ACM.

[47] P. Franco and E.J. McCluskey. On-line delay testing of digital circuits. In *VLSI Test Symposium, 1994. Proceedings., 12th IEEE*, pages 167–173, Apr 1994.

[48] Jeff George, Rocky Koga, Gary Swift, Greg Allen, Carl Carmichael, and Wei Tseng. Single event upsets in Xilinx Virtex-4 FPGA devices. In *Radiation Data Workshop of the Nuclear and Space Radiation Effects Conference*, pages 109–113, 2006.

[49] B. Gill, N. Seifert, and V. Zia. Comparison of alpha-particle and neutron-induced combinational and sequential logic error rates at the 32nm technology node. In *IEEE International Reliability Physics Symposium*, pages 199 –205, 26-30 2009.

[50] J. N. Glosli, K. J. Caspersen, J. A. Gunnels, D. F. Richards, R. E. Rudd, and F. H. Streitz. Extending stability beyond CPU millennium: A micron-scale atomistic simulation of kelvin-helmholtz instability. In *Proceedings ACM International Conference on Supercomputing*, 2007.

[51] J. N. Glosli, D. F. Richards, K. J. Caspersen, R. E. Rudd, J. A. Gunnels, and F. H. Streitz. Extending stability beyond CPU millennium: a micron-scale atomistic simulation of Kelvin-Helmholtz instability. In *Proceedings of the 2007 ACM/IEEE conference on Supercomputing*, pages 1–11, Reno, Nevada, 2007. ACM.

[52] Benjamin Gojman and André DeHon. Vmatch: Using logical variation to counteract physical variation in bottom-up, nanoscale systems. In *Proceedings of the International Conference on Field-Programmable Technology*, pages 78–87. IEEE, December 2009.

[53] O. Goloubeva, M. Rebaudengo, M. Sonza Reorda, and M. Violante. Soft-error detection using control flow assertions. In *Defect and Fault Tolerance in VLSI Systems, 2003. Proceedings. 18th IEEE International Symposium on*, pages 581–588, 2003.

[54] Sudhakar Govindavajhala and Andrew W. Appel. Using memory errors to attack a virtual machine. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2003.

[55] Shantanu Gupta, Shuguang Feng, Amin Ansari, Jason Blome, and Scott Mahlke. The stagenet fabric for constructing resilient multicore system. In *Proceedings of the International Symposium on Microarchitecture*, pages 141–151, 2008.

[56] Lance Hammond, Brian D. Carlstrom, Vicky Wong, Michael Chen, Christos Kozyrakis, and Kunle Olukotum. Transactional coherence and consistency: Simplifying parallel hardware and software. *IEEE Micro*, 24(6):92–103, November-December 2004.

[57] R. Harboe-Sorensen, F.-X. Guerre, and G. Lewis. Heavy-ion SEE test concept and results for DDR-II memories. *Nuclear Science, IEEE Transactions on*, 54(6):2125–2130, Dec. 2007.

[58] Reno Harboe-Sorensen, F.-X. Guerre, and G. Lewis. Heavy-ion SEE test concept and results for DDR-II memories. *IEEE Transactions on Nuclear Science*, 54(6):2125–2130, December 2007.

[59] R. Hauser, D. Hayes, L. Kallinen, D. Cannom, A. Epstein, A. Almquist, S. Song, G. Tyers, S. Vlay, and M. Irwin. Clinical experience with pacemaker pulse generators and transvenous leads: an 8-year prospective multicenter study. *Heart Rhythm*, 4(2):154–160, 2007.

[60] R. Hegde and N. R. Shanbhag. Soft digital signal processing. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 9(6):813–823, December 2001.

[61] D. F. Heidel, P. W. Marshall, J. A. Pellish, K. P. Rodbell, K. A. LaBel, J. R. Schwank, S. E. Rauch, M. C. Hakey, M. D. Berg, C. M. Castaneda, P. E. Dodd, M. R. Friendlich, A. D. Phan, C. M. Seidleck, M. R. Shaneyfelt, and M. A. Xapsos. Single-event upsets and multiple-bit upsets on a 45 nm SOI SRAM. *Nuclear Science, IEEE Transactions on*, 56(6):3499–3504, Dec. 2009.

[62] Henry Hoffmann, Sasa Misailovic, Stelios Sidiroglou, Anant Agarwal, and Martin Rinard. Using code perforation to improve performance, reduce energy consumption, and respond to failures. Computer Science and Artificial Intelligence Laboratory Technical Report MIT-CSAIL-TR-2009-042, MIT CSAIL, September 2009. Available online: `http://dspace.mit.edu/bitstream/handle/1721.1/46709/MIT-CSAIL-TR-2009-042.pdf`.

[63] M. Horowitz, E. Alon, D. Patil, S. Naffziger, Rajesh Kumar, and K. Bernstein. Scaling, power, and the future of CMOS. In *Technical Digest of the IEEE International Electron Device Meeting*, pages 7–15, December 2005.

[64] Kuang-Hua Huang and J. A. Abraham. Algorithm-based fault tolerance for matrix operations. *IEEE Trans. Comput.*, 33(6):518–528, 1984.

[65] D.B. Hunt and P.N. Marinos. General-purpose cache-aided rollback error recovery (CARER) technique. In *17th International Symposium on Fault-Tolerant Computing Systems*, pages 170–175. IEEE CS Press, 1987.

[66] L. D. Kannberg, M. C. Kintner-Meyer, D. P. Chassin, R. G. Pratt, J. G. DeSteese, L. A. Schienbein, S. G. Hauser, and W. M. Warwick. GridWiseTM: The benefits of a transformed energy system. Technical report, U.S. Department of Energy, September 2003.

[67] Jeffrey W. Kellington, Ryan McBeth, Pia Sanda, and Ronald N. Kalla. IBM POWER6 processor soft error tolerance analysis using proton irradiation. In *Proceedings of the IEEE Workshop on Silicon Errors in Logic – Systems Effects*, 2007. Available online: `ProcessorSoftErrorToleranceAnalysisUsingProtonIrradiation`.

[68] J. Kim and L. Kish. Error rate in current-controlled logic processors with shot noise. *Fluctuation and Noise Letters*, 4(1):83–86, 2004.

[69] Laszlo B. Kish. End of Moore's law: Thermal (noise) death of integration in micro and nano electronics. *Physics Letters A*, 305:144–149, 2002.

[70] Vijay Lakamraju and Russell Tessier. Tolerating operational faults in cluster-based FPGAs. In *Proceedings of the International Symposium on Field-Programmable Gate Arrays*, pages 187–194, 2000.

[71] P. Layton, S. Kniffin, S. Guertin, G. Swift, and S. Buchner. SEL induced latent damage, testing, and evaluation. *Nuclear Science, IEEE Transactions on*, 53(6):3153–3157, Dec. 2006.

[72] Xiaodong Li, Sarita V. Adve, Pradip Bose, and Jude A. Rivers. Architecture-level soft error analysis: Examining the limits of common assumptions. In *Proceedings of the International Conference on Dependable Systems and Networks*, pages 266–275, 2007.

[73] Yanjing Li, Young Moon Kim, Evelyn Mintarno, Donald S. Gardner, and Subhasish Mitra. Overcoming early-life failure and aging for robust systems. *Design & Test of Computers, IEEE*, 26(6):28–39, 2009.

[74] D.J. Lu. Watchdog processors and structural integrity checking. *Computers, IEEE Transactions on*, C-31(7):681–685, July 1982.

[75] R. E. Lyons and W. Vandekulk. The use of triple-modular redundancy to improve computer reliability. *IBM Journal of Research Development*, 6(2):200, 1962.

[76] Alain J. Martin. Towards an energy complexity of computation. *Information Processing Letters*, 77:181–187, 2001.

[77] Richard McCormack. DOD broadens "trusted" foundry program to include microelectronics supply chain. on web at http://www.allbusiness.com/government/government-bodies-offices-government/11420190-1.html.

[78] A. Meixner and D. J. Sorin. Detouring: Translating software to circumvent hard faults in simple cores. In *Proceedings of the 2008 IEEE International Conference on Dependable Systems and Networks With FTCS and DCC (DSN 2008)*, pages 80–89, 2008.

[79] Albert Meixner, Michael E. Bauer, and Daniel Sorin. Argus: Low-cost, comprehensive error detection in simple cores. In *Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 210–222. IEEE Computer Society, 2007.

[80] R. Melhem and R.Graybill, editors. *Power-Aware Computing*, chapter ET2: A Metric For Time and Energy Efficiency of Computation. Kluwer Academic Publishers, 2001.

[81] Cross-Layer Resilience Challenges: Metrics and Optimization. Subhasish mitra and kevin brelsford and pia n. sanda. In *Proceedings of the Conference and Exhibition on Design, Automation and Test in Europe*, 2010.

[82] Shubhendu S. Mukherjee, Michael Kontz, and Steven K. Reinhardt. Detailed design and evaluation of redundant multithreading alternatives. *SIGARCH Comput. Archit. News*, 30(2):99–110, 2002.

[83] Shubhendu S. Mukherjee, Christopher T. Weaver, Joel Emer, Steven K. Reinhardt, and Todd Austin. Measuring architectural vulnerability factors. *IEEE Micro*, 23(6):70–75, November–December 2003.

[84] Nicola Muscettola, P. Pandurang Nayak, Barney Pell, and Brian C. Williams. Remote agent: To boldly go where no AI system has gone before. *Artificial Intelligence*, 103((1-2)):5–48, August 1998.

[85] Sani R. Nassif, Nikil Mehta, and Yu Cao. A resilience roadmap. In *Proceedings of the Conference and Exhibition on Design, Automation and Test in Europe*, 2010.

[86] John Von Neumann. Probabilistic logic and the synthesis of reliable organisms from unreliable components. In Claude Shannon and John McCarthy, editors, *Automata Studies*. Princeton University Press, 1956.

[87] Borivoje Nikolic. Design in the power-limited scaling regime. *IEEE Transactions on Electron Devices*, 55(1):71–83, Nanuary 2008.

[88] OCWorkbench.com. Benchmarks of ECS 945GCT-D with intel atom 1.6ghz. http://my.ocworkbench.com/2008/ecs/ECS_945GCT-D_Atom_board/b1.htm.

[89] U.S. Bureau of Transporation Statistics. Number of u.s. aircraft, vehicles, vessels, and other conveyances. on web at http://www.bts.gov/publications/national_transportation_statistics/html/table_01_11.html, July 2010.

[90] N. Oh, S. Mitra, and E.J. McCluskey. ED^{4}I: error detection by diverse data and duplicated instructions. *Computers, IEEE Transactions on*, 51(2):180–199, Feb 2002.

[91] N. Oh, P.P. Shirvani, and E.J. McCluskey. Error detection by duplicated instructions in super-scalar processors. *Reliability, IEEE Transactions on*, 51(1):63–75, Mar 2002.

[92] Sung-Boem Park and Subhasish Mitra. IFRA: instruction footprint recording and analysis for post-silicon bug localization in processors. In *Proceedings of the 45th annual Design Automation Conference*, pages 373–378, Anaheim, California, 2008. ACM.

[93] Karthik Pattabiraman, Zbigniew Kalbarczyk, and Ravishankar K. Iyer. Automated derivation of application-aware error detectors using static analysis. In *Proceedings of the 13th IEEE International On-Line Testing Symposium*, pages 211–216. IEEE Computer Society, 2007.

[94] Andrea Pellegrini, Valeria Bertacco, and Todd Austin. Fault-based attack of rsa authentication. In *Proceedings of the Conference and Exhibition on Design, Automation and Test in Europe*, 2010.

[95] Nicholas Pippenger. Developments in 'the synthesis of reliable organisms from unreliable components'. In *Proceedings of the Symposia of Pure Mathematics*, volume 50, pages 311–324, 1990.

[96] James S. Planck, Micha Beck, Gerry Kingsley, and Kai Li. Libckpt: Transparent checkpointing under UNIX. In *Usenix*, pages 213–223, New Orleans, LA, 1995.

[97] Michael D. Powell, Arijit Biswas, Shantanu Gupta, and Shubhendu S. Mukherjee. Architectural core salvaging in a multi-core processor for hard-error tolerance. In *Proceedings of the 36th annual International Symposium on Computer Architecture*, pages 93–104, Austin, TX, USA, 2009. ACM.

[98] M. Prvulovic, Zhang Zheng, and J. Torrellas. ReVive: cost-effective architectural support for rollback recovery in shared-memory multiprocessors. In *Computer Architecture, 2002. Proceedings. 29th Annual International Symposium on*, pages 111–122, 2002.

[99] Heather Quinn, Paul Graham, Jim Krone, Michael Caffrey, and Sana Rezgui. Radiation-induced multi-bit upsets in SRAM-based FPGAs. *IEEE Transactions on Nuclear Science*, 52(6):2455 – 2461, December 2005.

[100] Heather Quinn, Keith Morgan, Paul Graham, Jim Krone, and Michael Caffrey. Static proton and heavy ion testing of the Xilinx Virtex-5 device. In *The Proceedings of Data Workshop for Nuclear and Space Radiation Effects Conference*, pages 177 – 184, July 2007.

[101] Joydeep Ray, James C. Hoe, and Babak Falsafi. Dual use of superscalar datapath for transient-fault detection and recovery. In *Proceedings of the 34th annual ACM/IEEE International Symposium on Microarchitecture*, pages 214–224, Austin, Texas, 2001. IEEE Computer Society.

[102] Vijay Janapa Reddi, Meeta S. Gupta, Glenn Holloway, Michael D. Smith, Gu-Yeon Wei, and David Brooks. Voltage emergency prediction: A signature-based approach to reducing voltage emergencies. In *Proceedings of the International Symposium on High-Performance Computer Architecture*, 2009.

[103] K. Reick, P.N. Sanda, S. Swaney, J.W. Kellington, M.J. Mack, M.S. Floyd, and D. Henderson. Fault-tolerant design of the IBM Power6 microprocessor. *Micro, IEEE*, 28(2):30–38, March-April 2008.

[104] Steven K. Reinhardt and Shubhendu S. Mukherjee. Transient fault detection via simultaneous multithreading. *SIGARCH Comput. Archit. News*, 28(2):25–36, 2000.

[105] George A. Reis, Jonathan Chang, Neil Vachharajani, Ram Rangan, David I. August, and Shubhendu S. Mukherjee. Software-controlled fault tolerance. *ACM Trans. Archit. Code Optim.*, 2(4):366–396, 2005.

[106] Shing-Hwa Renn, Christine Raynaud, Jean-Luc Pelloie, and Francis Balestra. A thorough investigation of the degradation induced by hot-carrier injection in deep submicron n- and p-channel partially and fully depleted unibond and SIMOX MOSFETs. *IEEE Transactions on Electron Devices*, 45(10):2146–2152, October 1998.

[107] John Rhea. BAE systems moves into third generation rad-hard processors. Military and Aerospace Electronics, `http://www.militaryaerospace.com/index/display/article-display/143464/articles/military-aerospace-electronics/volume-13/issue-5/news/bae-systems-moves-into-third-generation-rad-hard-processors.html`.

[108] Bogdan F. Romanescu and Daniel J. Sorin. Core cannibalization architecture: improving lifetime chip performance for multicore processors in the presence of hard faults. In *Proceedings of the 2008 International Conference on Parallel Architectures and Compilation Techniques (PACT)*, pages 43–51. ACM, 2008.

[109] Eric Rotenberg. Ar-SMT: A microarchitectural approach to fault tolerance in microprocessors. In *Proceedings of the Twenty-Ninth Annual International Symposium on Fault-Tolerant Computing*. IEEE Computer Society, 1999.

[110] Raphael Rubin and André DeHon. Choose-Your-Own-Adventure Routing: Lightweight Load-Time Defect Avoidance. In *Proceedings of the International Symposium on Field-Programmable Gate Arrays*, pages 23–32, 2009.

[111] S. K. Sahoo, Li Man-Lap, P. Ramachandran, S. V. Adve, V. S. Adve, and Zhou Yuanyuan. Using likely program invariants to detect hardware errors. In *Proceedings of the 2008 IEEE International Conference on Dependable Systems and Networks With FTCS and DCC (DSN 2008)*, pages 70–79, 2008.

[112] John E. Savage. Area-time tradeoffs for matrix multiplication and related problems in VLSI models. *Journal of Computer and Systems Sciences*, 22(2):230–242, 1981.

[113] J. Sayer, D LeBlanc, S Bogard, E Nodine, and W. Najm. Integrated vehicle-based safety systems (IVBSS), third annual report. National Highway Safety Administration Report, DOT HS 811 221, 2009.

[114] Dieter K. Schroder and Jeff A. Babcock. Negative bias temperature instability: Road to cross in deep submicron silicon semiconductor manufacturing. *Journal of Applied Physics*, 94(1):1–18, July 2003.

[115] Ethan Schuchman and T. N. Vijaykumar. Rescue: A microarchitecture for testability and defect tolerance. In *Proceedings of the 32nd annual International Symposium on Computer Architecture*, pages 160–171. IEEE Computer Society, 2005.

[116] Adi Shamir. Research announcement: Microprocessor bugs can be security disasters. Available online at `http://cryptome.org/bug-attack.htm`, November 2007.

[117] Claude E. Shannon and Warren Weaver. *The Mathematical Theory of Communication*. University of Illinois Press, 1949.

[118] Nir Shavit and Dan Touitou. Software transactional memory. In *Proceedings of the ACM symposium on Principles of Distributed Computing*, pages 204–213, 1995.

[119] B. Shim and N. R. Shanbhag. Energy-efficient soft-error tolerant digital signal processing. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 14(4):336–348, April 2006.

[120] Smitha Shyam, Kypros Constantinides, Sujay Phadke, Valeria Bertacco, and Todd Austin. Ultra low-cost defect protection for microprocessor pipelines. In *Proceedings of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 73–82, San Jose, California, USA, 2006. ACM.

[121] Stelios Sidiroglou, Oren Laadan, Angelos D. Keromytis, and Jason Nieh. Using rescue points to navigate software recovery. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 273–280, 2007.

[122] A.L. Silburt, A. Evans, I. Perryman, Shi-Jie Wen, and D. Alexandrescu. Design for soft error resiliency in internet core routers. *Nuclear Science, IEEE Transactions on*, 56(6):3551–3555, December 2009.

[123] D. J. Sorin, M. M. K. Martin, M. D. Hill, and D. A. Wood. SafetyNet: improving the availability of shared memory multiprocessors with global checkpoint/recovery. In *Computer Architecture, 2002. Proceedings. 29th Annual International Symposium on*, pages 123–134, 2002.

[124] Vilas Sridharan and David R. Kaeli. Eliminating microarchitectural dependency from architectural vulnerability. In *HPCA*, pages 117–128, 2009.

[125] Jayanth Srinivasan, Sarita V. Adve, Pradip Bose, and Jude A. Rivers. The impact of technology scaling on lifetime reliability. In *Proceedings of the 2004 International Conference on Dependable Systems and Networks*, pages 177–186, 2004.

[126] Gary M. Swift. Virtex-II static SEU characterization. Technical Report 1, Xilinx Radiation Test Consortium, 2004.

[127] BAE Systems. http://www.baesystems.com/BAEProd/groups/public/@businesses/@eandis/documents/bae_publication/bae_pdf_eis_rad750_pwr_pc_mp.pdf.

[128] C. Thompson. Area-time complexity for VLSI. In *Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing*, pages 81–88, May 1979.

[129] Rajesh Venkatasubramanian, J. P. Hayes, and B. T. Murray. Low-cost on-line fault detection using control flow assertions. In *On-Line Testing Symposium, 2003. IOLTS 2003. 9th IEEE*, pages 137–143, 2003.

[130] S. N. Vernov, E. V. Gorchakov, P. I. Shavrin, and K. N. Sharvina. Radiation belts in the region of the south-atlantic magnetic anomaly. *Space Science Review*, 7:490–533, 1967.

[131] Nicholas Wang and Sanjay Patel. ReStore: Symptom based soft error detection in microprocessors. In *Proceedings of the 2005 International Conference on Dependable Systems and Networks*, pages 30–39. IEEE Computer Society, 2005.

[132] C. Wilkerson, Hongliang Gao, A.R. Alameldeen, Z. Chishti, M. Khellah, and Shih-Lien Lu. Trading off cache capacity for reliability to enable low voltage operation. In *Proceedings of the International Symposium on Computer Architecture*, pages 203–214, June 2008.

[133] Chris Wilkerson, Hongliang Gao, Alaa R. Alameldeen, Zeshan Chishti, Muhammad Khellah, and Shih-Lien Lu. Trading off cache capacity for reliability to enable low voltage operation. *SIGARCH Comput. Archit. News (ISCA 2008)*, 36(3):203–214, 2008.

[134] Jun Xu, Shuo Chen, Zbigniew Kalbarczyk, and Ravishankar K. Iyer. An experimental study of security vulnerabilities caused by errors. In *Proceedings of International Conference on Dependable Systems and Networks*, pages 421–432, 2001.

[135] Y. C. (Bob) Yeh. Triple-triple redundant 777 primary flight computer. In *Proceedings of the Aerospace Applications Conference*, pages 293–307, 1996.

[136] Mahmut Yilmaz, Derek R. Hower, Sule Ozev, and Daniel J. Sorin. Self-checking and self-diagnosing 32-bit microprocessor multiplier. In *Proceedings of International Test Conference*, 2006.

[137] Anthony J. Yu and Guy G. Lemieux. FPGA defect tolerance: Impact of granularity. In *Proceedings of the International Conference on Field-Programmable Technology*, pages 189–196, 2005.

[138] Ming Zhang, Subhasish Mitra, T. M. Mak, Norbert Seifert, Nicholas J. Wang, Quan Shi, Kee Sup Kim, Naresh R. Shanbhag, and Sanjay J. Patel. Sequential element design with built-in soft error resilience. *IEEE Trans. Very Large Scale Integr. Syst.*, 14(12):1368–1378, 2006.

A Process and Participants

A.1 Process

The process for the Cross-Layer Reliability Study included:

- Forming a core working group to serve as a steering committee for the study
- Organizing and holding three public workshops that allowed the organizers, core working group, constituency group members, and participants to have face-to-face meetings to discuss reliability challenges,
- Forming constituency groups that provided input into reliability challenges based on particular fields; these met regularly via teleconferences and briefed at the second and third workshops,
- Maintaining a wiki page to allow for open communication and opportunities for people to contribute outside of the workshops,
- Presenting study ideas at a number of conferences to discuss the study openly with the larger research community, and
- Contributing to the International Technology Roadmap for Semiconductors.

Details on these aspects of the process will be discussed in the remaining part of this section.

A.1.1 Study Participants and Contributors

The core working group was an important aspect of our process. The core working group members included academic and industry researchers who had a history of working in the reliability and computation fields. These members provided a broad base of expertise to identify and invite participants, review documents before release, review workshop planning and digest workshop results, and provide outreach to the larger reliability research community.

The constituency groups also played a key role in our process. During the first workshop, it was decided that the need for reliability and the amount of overhead spent on reliability was dependent on the system. In the second and third meetings we relied upon briefings from five constituency groups (Aerospace, Life-Critical, Large-Scale, Consumer, and Infrastructure) for guidance about how reliability affected different types of systems. Constituency group members were identified through the core working group, the study organizers, and the group leaders. Many of these groups met weekly and/or at other conferences to provide briefings to the study workshops. They also gathered at the study meetings to respond to questions from the study participants and evolve their challenges based on interactions with other groups. Because of the work of the constituency groups, we were able to find strong trends that affected multiple constituency groups (Section 4.2). For example, the life-critical, infrastructure, and aerospace groups all agreed that the reliability of passives and analog devices were a challenge for their systems. Finally, the constituency groups provided inputs for this final report (Appendix E).

Beyond the constituency groups, there were also two subgroups that developed out of the first workshop that were interested in metrics and roadmapping. Both groups briefed at both the second and third meetings and had papers at Design Automation and Test in Europe (DATE 2010)

[85, 81]. The roadmapping group also contributed to the International Technology Roadmap for Semiconductors [7, Design Chapter, pp. 27–28, including Figure DESN8].

A.1.2 Workshops

We held three public workshops. Information about the meetings was published on our wiki page so that people could contact us if they were interested.

The first workshop was co-located with the 2009 IEEE Workshop on Silicon Errors in Logic - System Effects (SELSE) at Stanford University. Consulting with the core working group and the SELSE organizers, we invited a large number of industry representatives to participate in this workshop and help identify the key agenda.

Review of the first meeting by the core working group led to the formation of the constituency and focus groups. We recruited leaders for these groups from the active participants at the first workshop. Constituency and focus group leaders pulled in additional participants based on the expertise and coverage they needed. Participation in these groups, in turn, determined invitations for the second and third workshops, which were held at Los Alamos, NM in July of 2009 and Austin, TX in October of 2009. The second workshop focused on refining our understanding of the key challenges in reliability and how different segments of the industry were affected by reliability challenges, while the third workshop concentrated on the study's conclusions and recommendations.

A.1.3 Wiki Page and Summary Reports

During the study, a wiki was created at `www.relxlayer.org` to host discussions in-between studies, reports, and information about the workshops. The wiki page has been online since February of 2009. The wiki allowed open discussion of reliability topics and an opportunity for the larger reliability community to discuss the Cross-Layer Reliability Study, whether they were participants in the workshops or not. The wiki page has summaries and presentations from the three workshops, and we refer the interested reader to those summaries rather than reproducing them here.

A.1.4 Public Discussion of Study Results

The final aspect of the study that the organizers and core working group members have been participating in is presenting results of the three workshops at other conferences. At this stage we have presented the study at these conferences:

- A special session on the Cross-Layer Reliability study was held at Design Automation and Test in Europe (DATE 2010). During this session, papers on the reliability roadmap, the Cross-Layer vision, Cross-Layer techniques, and metrics were presented.
- A talk was given at the 2010 IEEE Workshop on Silicon Errors in Logic - System Effects at Stanford University. The talk presented the Cross-Layer study and some of the Cross-Layer techniques.
- A plenary talk was given at Space Computing 2010. The talk gave an overview of the Cross-Layer Study and presented the aerospace challenges to the participants.
- A presentation and a birds of a feather session on Cross-Layer reliability was held at Dependable Systems and Networks 2010.

A.2 Participants

- Sarita Adve, University of Illinois at Urbana-Champaign
- Marcos Aguilera, Microsoft Research
- Carl Anderson, IBM
- Paul Armijo, General Dynamics Advanced Information Systems
- Todd Austin, University of Michigan
- Sankar Basu, NSF
- Lori Bechtold, Boeing Research & Technology
- Shawn Blanton, Carnegie Mellon University
- Shekhar Borkar, Intel Corporation
- Younes Boulghassoul, Information Sciences Institute - University of Southern California
- Keith Bowman, Intel Corporation
- Greg Bronevetsky, LLNL
- James Browne, University of Texas Austin
- Nicholas Carter, Intel Corporation
- Vikas Chandra, ARM
- Tim Cheng, University of California Santa Barbara
- Pierre Chor-Fung Chia, Cisco Systems Inc.
- Lewis Cohn, NRL
- John Daly, Department of Defense
- Chitaranjan Das, NSF
- J.L. de Jong, Xilinx
- Nathan DeBardeleben, LANL
- Erik deBenedictis, Sandia National Laboratories
- André DeHon, University of Pennsylvania
- Eliezer Dekel, IBM
- Bill Eklow, Cisco Systems Inc.
- Glenn A Forman, General Electric
- Armando Fox, University of California Berkeley
- Tim Gallagher, Lockhead Martin Corporation
- Donald S. Gardner, Intel Corporation
- Kinshuk Govil, VMware
- John Gustafson, Intel Corporation
- Eric Hannah, Intel Corporation
- William Harrod, DARPA
- William Heidergott, General Dynamics Advanced Information Systems
- John Hiller, STA
- Andrew Huang, Bunnie Studios LLC Chumby Industries
- Ravi Iyer, University of Illinois at Urbana-Champaign
- David Kaeli, Northeastern University
- Zbigniew Kalbarczyk, University of Illinois at Urbana-Champaign
- Kevin Kemp, Freescale
- Prabhakar Kudva, IBM
- Kimmo Kuusilinna, Nokia
- Shih-Lien Lu, Intel Corporation

- James Lyke, Air Force Research Laboratory
- William M. Jones Jr, Coastal Carolina University
- Nikil Mehta, California Institute of Technology
- Sarah Michalak, LANL
- Subhasish Mitra, Stanford University
- Claude Moughanni, Freescale Semiconductor
- Shubu Mukherjee, Intel Corporation
- Helia Naeimi, Intel Corporation
- Sani Nassif, IBM
- Suriyaprakash Natarajan, Intel Corporation
- Eugene Normand, Boeing Research & Technology
- Kevin Nowka, IBM
- Ishwar Parulkar, Cisco Systems
- Karthik Pattabiraman, Microsoft
- Mark Porter, Medtronic
- Heather Quinn, LANL
- Charles Recchia, Intel Corporation
- Anthony Reipold, Freescale Semiconductor
- Pia Sanda, IBM
- Sumeet Sandhu, Intel Corporation
- John Savage, Brown University
- Bianca Schroeder, University of Toronto
- Sanjit Seshia, UC Berkeley
- Allan Silburt, Cisco Systems
- James Smith, Intel Corporation
- Rafi Some, JPL
- Daniel Sorin, Duke University
- Jon Stearley, Sandia
- Gary Swift, Xilinx Inc.
- David Tennenhouse, New Venture Partners
- Chandra Tirumurti, Intel Corporation
- Steve Trimberger, Xilinx
- Ian Troxel, SEAKR Engineering Inc.
- David Walker, Princeton University
- Shi-Jie Wen, Cisco Systems Inc.
- Chris Wilkerson, Intel Corporation
- Alan Wood, Sun Microsystems
- Vivian Zhu, Texas Instruments

B Sample Solicitation

B.1 Introduction

Advances in fabrication processes are creating a tension between reliability and efficiency. Increasing rates of faults, variation, and aging in integrated circuits are forcing engineers to assume that devices and circuits may not perform as designed. At the same time, power has become the dominant constraint on design complexity, making it critical that designs minimize the amount of energy they spend tolerating non-ideal device and circuit behavior. Current reliability techniques, such as voltage/clock rate margins, replication, and disk-based checkpointing will not be able to satisfy the competing requirements for future integrated circuits. These reliability techniques typically operate at one level of the system stack, which forces them to make worst-case assumptions about the other layers in the stack, leading to inefficiencies that will make them impractical in future fabrication processes. For example, hardware-only error-correction techniques have no information about an application's sensitivity to errors, and thus cannot take advantage of applications such as video playback that can tolerate errors in some of their computations without significant impact on their results. Similarly, techniques that have no information about a system's condition and operating environment must assume worst-case error rates, temperature, aging, and variation, leading to significant inefficiency.

Cross-layer approaches to reliability have the potential to deliver high reliability with significantly lower power and performance overheads than current single-layer techniques. By distributing reliability across the system stack, these approaches can take advantage of the information available at each level to efficiently tolerate errors, aging, and variation, can handle different physical effects at the most efficient stack layer, and can adapt to varying application needs, operating environments, and changing hardware state. This solicitation requests proposals for research that will develop and define the emerging area of cross-layer reliability.

B.2 Research Solicited

Proposals are solicited for research into techniques, tools, and metrics for cross-layer approaches to reliability. Topics of interest include, but are not limited to:

- **Self-repairing systems:** techniques to diagnose changing system state over time, interfaces to expose and control reconfiguration across stack layer boundaries, and architectures that support repair by isolating defective components and eliminating single-points-of-failure
- **Multi-level error filtering:** analyses of the costs of filtering different types of errors at different levels of the stack, techniques to communicate the types and rates of errors that each level can handle, and approaches that allow systems to dynamically select which level of the stack handles a given type of error
- **Interfaces for multi-level cooperation:** abstractions that hide unnecessary details from other layers of the stack while communicating critical information about reliability and errors, interfaces that allow software to communicate reliability needs, invariants, and capabilities to hardware, and techniques that allow hardware to communicate its state to software without requiring that the software understand the details of the hardware design

- **Light-weight checking and application-level reliability:** analyses of classes of computations that are amenable to self-checking and programming methodologies that support self-checking and self-correcting code
- **Differential reliability:** techniques that allow applications and hardware devices to express how much reliability they need and what errors they can tolerate to support system-level power/performance/reliability optimization
- **Scalable adaptive reliability:** approaches that allow systems to tailor their reliability to the needs of end users, the constraints imposed by different operating environments, and changing system state
- **Graceful degradation and adaptation:** approaches that allow systems to adapt to provide the best possible performance, reliability, and quality of service in the face of changing (usually degrading) system capabilities. Techniques that express an application's sensitivity to parallelism and clock rate so that the system can optimize for performance/energy are of particular interest.
- **Standard models for cross-layer reliable systems:** designs for base cross-layer reliable systems that other researchers can extend and modify. Proposals for research in this area should include a discussion of how the models will be made available to other researchers.
- **Tools and automation for cross-layer reliable design:** simulators, modeling frameworks, reliability analysis tools, and optimization tools. Proposals on these topics should include a plan to release the developed software to the research community via a free or open-source license.
- **Metrics for reliable design:** techniques to evaluate the effectiveness of reliability schemes, characterize costs, and guide design and optimization.

Proposals should clearly describe how the proposed research will extend across multiple layers of the system stack or will enable cross-layer optimizations. Proposals should also include a plan to quantify the cost-benefit trade-offs of the techniques under investigation.

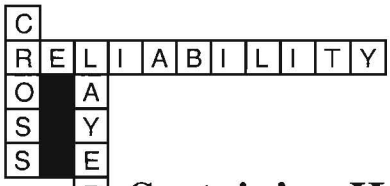
B.3 Program Structure

Proposals are solicited for small (1-3 investigators) and medium (4-6 investigators) proposals. Submissions from cross-disciplinary or multi-institution teams are encouraged. Programs funded under this solicitation will have a duration of 36 months. Small proposals should have a maximum budget of $X, while medium proposals have a maximum budget of $Y.

To encourage collaboration between research efforts, program meetings will be held every six months, and proposals are expected to budget sufficient travel funds to allow key personnel to attend these meetings. As a further encouragement, there will be an opportunity after each of the first four program meetings to apply for additional funding to support cross-team collaborative efforts or to support public releases of tools developed as part of this program.

C Non-Technical Executive Summary

Following is a less-technical summary of the study and recommendations.



ImmunoLogics: Sustaining Healthy Growth of the Silicon Economy

(Non-Technical Executive Summary of the CCC Visioning Study on Cross-Layer Reliability)

ImmunoLogics *The study of layered computer systems designed to identify and neutralize hardware errors. Analogous to the human immune system that has innate and adaptive systems to provide immediate and long-term responses to invading pathogens.*

Problem As our computer systems grow in size and capabilities while transistors, which are the most basic element of computation, continue to shrink, they become increasingly prone to failure and early death. Over the past half century, shrinking transistors have allowed the capacity and speed of computer systems to double roughly every two years, a phenomenon known as Moore's Law. This amazing trend has enabled incredible American economic innovation, including Google Search, Photoshop, iPods, Facebook, E-banking, Smart Phones, personal GPSes, digital video recorders, and many, many others. However, these tiny transistors are now at a breaking point: if we make them much smaller, they become noticeably less reliable. Today's computers can too easily fail or die when just a few transistors fail, and tomorrow's transistors will be even more likely to fail. If we hope to enable the next round of American innovation in computer systems and consumer electronics, we must find ways for our systems to cope with failing transistors.

Unlike today's computers, biological systems can tolerate dead or misbehaving cells, eliminate them, and actively respond to pathogens that attempt to damage the organism. The key to this robustness is an immune system that provides multiple layers of response to upsets to the biological system. Today's computer and electronic systems are fragile precisely because they lack a robust immune system to tolerate and repair damage. The lack of an immune system was tolerable when transistors were large and reliable and when computers were not so widely used for critical functions. However, as computational capacity grows and becomes the glue that supports our modern infrastructure, frequent failure and manual repair cease to be viable options. *We must develop an automated immune system for computations to allow further transistor scaling and guarantee that our computers, communication, and automation are not incapacitated at inopportune times.*

Tomorrow's smaller transistors have a higher likelihood of misbehaving than their larger and slower predecessors, causing computers to burn excessive energy, heat up, slow down, age rapidly, fail often, and die early. In the past, these symptoms were mild, and we could treat them with brute-force remedies, including safety margins and replication. However, as symptoms become more severe, it will cost too much energy to stabilize devices using traditional remedies, and we risk unreasonably high rates of failure and wearout. Even today, these effects cause one of the world's largest supercomputers to crash multiple times per day.[3] Recovering from these crashes reduces the work it can perform by over 20%. Alternately, an immune-system-like response strategy that works at multiple levels to efficiently detect failures, adapt to changes, and reconfigure around unhealthy components to effect repairs provides a more promising path to support larger, more capable, and more trustworthy computing systems. At the lowest level, transistors are expected to fail. At intermediate levels, innate systems rapidly detect misbehavior and recover the computation. Rather than demanding that a single layer or mechanism catch all errors, multiple layers and mechanisms cooperate, each catching the problems that are easiest for it to detect. Meanwhile, higher level systems learn the best way to adapt the computation for long-term health and efficiency.

[3] 131,072 CPU BlueGene/L at Lawrence Livermore National Laboratory–*Supercomputing '07*

Why it Matters Moore's Law transistor size reduction and increasing integrated circuit capabilities have been an engine of growth for the U.S. economy, enabling new products and services, creating new value and wealth, increasing safety, and removing menial tasks from our daily lives. The silicon health problems identified above could spell the end of transistor scaling, depriving us of further economic, safety, and quality-of-life benefits. The U.S. has consistently created and monetized new value in computing and automation technology over the past several decades. The multi-level mitigation techniques discussed above are essential if we are to continue to reap the benefits of Moore's Law, including the continued value creation that energizes and sustains the U.S. economy.[4] Furthermore, U.S. health, transportation, finance, commerce, intelligence, and military superiority all rely heavily on harnessing advancing computing technology.

Recommendations We recommend the support of a robust, long-term research and education effort to invent, develop, and refine ImmunoLogics systems. Specific components of this research effort include understanding:

1. How do we design hardware and software organizations that are prepared for repair?
2. What is the right amount of error filtering at each level in the stack—from circuits through application software—and what are the best techniques for filtering?
3. How do we formulate, analyze, and manage multilevel trade-offs for fault mitigation that generalize the idea of hardware-software trade-offs, including the interfaces for cross-layer information sharing?
4. What is a theory and design pattern set for efficient, light-weight error checking that exploits the high-level properties of hardware architectures, software applications, and algorithms?
5. What is a theory and practical framework for expressing and reasoning about differential reliability, including both application needs and hardware/system organization to meet those needs?
6. How do we design scalable system solutions that can adapt to varying error rates and reliability demands?
7. How do we design components and systems that degrade gracefully and systems that are aware of their overall health?

An education component would address embedding reliability and ImmunoLogics engineering into Electrical Engineering, Computer Engineering, and Computer Science curricula.

The Role of the U.S. Government U.S. Government leadership is necessary for this important work to move forward. Robust ImmunoLogics responses cross the entire computing system stack from integrated circuits to software applications. With today's horizontal companies, no one vendor or research laboratory can effect this revolutionary change alone. Furthermore, since an individual consumer cannot reasonably assess the risks and benefits of today's highly complex computer systems, normal market forces alone are insufficient to produce solutions that adequately ensure public safety. Industry-wide standards and safety ratings[5] can give consumers and integrators the insight they need into the resilience of complex silicon technology and incentivize responsible industry development of trustworthy technology.

Acknowledgment This material is based upon work supported by the National Science Foundation (NSF) under Grant 0637190 to the Computing Research Association (CRA) for the Computing Community Consortium (CCC). The opinions, findings, conclusions, and recommendations expressed in this material are those of the study participants and do not necessarily reflect the views of the NSF, CRA, CCC, or the employers of the study leadership or participants.

[4]Consumer Electronics alone contributes 5–10% of the U.S. GDP according to the Consumer Electronics Association <http://www.ce.org/PDF/CEA_Final_Report_20080401_Lo-Res.pdf>

[5]*c.f.* 5-star automotive crash ratings

Technology	σ_L (nm)	$\sigma_{V_{th}}$ (mV)	σ_μ (percent)
45nm	4.5	17.5	4.3
32nm	3.4	24.0	6.1
22nm	2.2	32.3	8.6
16nm	1.6	37.2	11
12nm	1.2	43.7	13.2

Table 3: Standard Deviation of Device Parameter Variation for $W/L = 8$ Inverter Circuit

D Roadmap

One topic that came up many times during the study was the lack of good, publicly-available, predictions of variation and error rates in future fabrication processes. While it is very difficult to accurately predict the characteristics of far-future devices, good models can help researchers decide which areas to target by identifying trends and providing estimates of the relative importance of different factors. To address this problem, Sani Nassif, Nikil Mehta, and Yu Cao organized a subgroup of the study participants to model the impact of variation on error rates in different types of circuits for future fabrication processes. This appendix summarizes that work, which was published in [85] and has been incorporated into the International Technology Roadmap for Semiconductors [7, Design Chapter, pp. 27–28, including Figure DESN8].[6]

D.1 Approach

Starting with the Predictive Technology Models maintained at Arizona State University (http://ptm.asu.edu), the team estimated the probability distribution of three device parameters that are affected by variation: device length (L), threshold voltage (V_{th}), and charge mobility μ. They then calculated the predicted standard deviation σ of each of these parameters in 45nm, 32nm, 22nm, 16nm, and 12nm fabrication processes, the results of which are shown in Table 3.

This analysis predicts that device length variation will remain effectively constant at approximately 10% of total device length, while the percentage variation in charge mobility will triple as we approach 12nm processes. More importantly, it predicts that the absolute variation in threshold voltage will increase significantly in future processes, which is a major concern given the nonlinear dependence of drive current on threshold voltage and the desire to reduce threshold voltages in order to reduce supply voltages.

To translate parameter variation into estimates of circuit failure rates, the team studied three basic circuits: inverters, SRAM cells, and latches. For each circuit, they determined the direction of variation for each of the three device parameters that led to the greatest degradation in circuit performance, and then increased the variation of each parameter by the same number of standard deviations until the circuit failed. This gave the estimate of the number of σ of variation in the worst-case direction that the circuit could sustain before failing, which is shown in Figure 5 and reproduced here as Figure 21.

As an example, the worst-case direction of variation for the inverters studied was the direction that made the P device strongest and the N device weakest, leading to failure when the "on" current of the N device was comparable to the "off" current through the P device. When this occurred,

[6]DESN8 is an earlier and less accurate version of Figure 21.

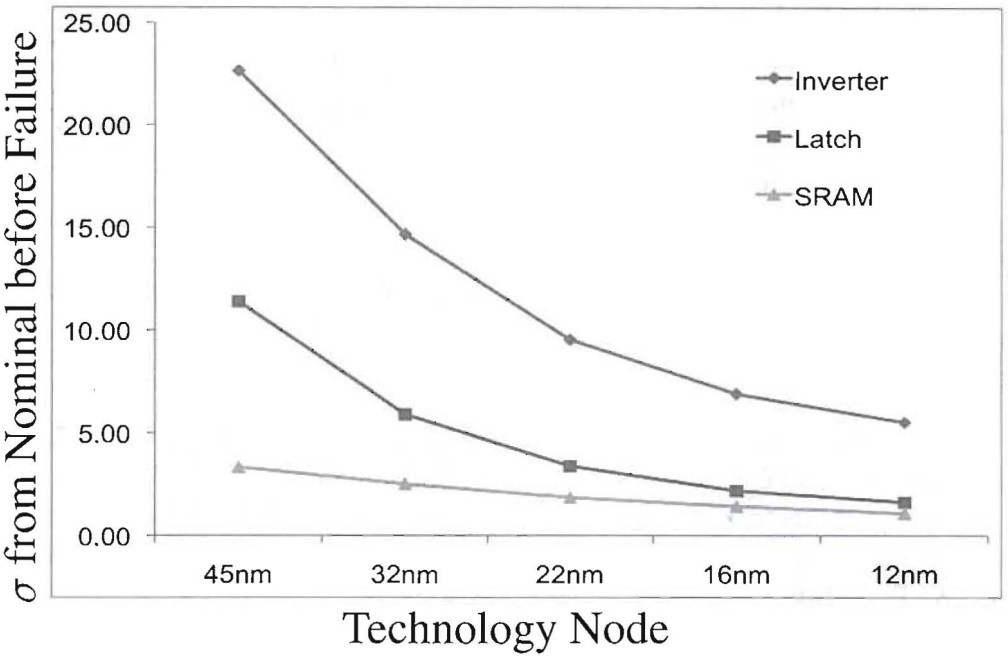


Figure 21: Failure rate per circuit element due to variation effects

the N device was unable to pull the output voltage down to a valid low value, making the inverter unable to drive its output devices. In the simulated 45nm process, inverters remained functional out to 22.6 standard deviations of variation in the worst-case direction. In the simulated 12nm process, inverters failed at 5.53 σ of worst-case variation, comparable to the failure point for latches in the simulated 32nm process.

Qualitatively, these results show several important trends. Variation-induced hard failure rates in SRAM devices (i.e., neglecting failures caused by fabrication defects) are expected to increase going forward, but at a relatively low rate. Variation-induced failures in latches are expected to increase much more quickly, becoming comparable to current (45nm) SRAM failure rates between the 22nm and 16nm nodes. Inverter failure rates due to variation will also increase significantly, with 22nm and subsequent processes having higher variation-induced failure rates than latches in 45nm processes. These results are based on complete failure. Long before this point, the circuits will typically be unusably slow, so the practically important failure rates will be even higher.

Since the high-level goal of this effort was to develop a publicly-available roadmap for reliability issues in CMOS processes, the roadmap team has made all of their scripts and models publicly available, and encourages community participation in further refining their work.

URL: <FIXME:TODO: proper URL is at ASU – still need to get from Nikil>

Please contact Sani Nassif (sani.nassif@gmail.com) for further information.

E Constituency Group Reports

E.1 Aerospace

Authors: *Heather Quinn (LANL), Lew Cohen (Navy Research Laboratory), Tim Gallagher (Lockheed-Martin), Paul Armijo (General Dynamics), Ian Troxel (SEAKR), Rafi Some (JPL), Jim Lyke (AFRL-Kirtland), Eugene Normand (Boeing), Lori Bechtold (Boeing) , Erik DeBenedictis (Sandia), and David Walker (Princeton)*

Microelectronics used for aerospace applications, whether for aircraft or spacecraft, are subject to a variety of stringent requirements that include, but are not limited to:

- Significant size, weight and power (SWaP) constraints.
- Extreme reliability requirements in often harsh operating environments such as radiation, temperature, humidity, shock, or vibration. For example, many systems must be able to operate in the natural space or upper atmosphere radiation environment and many military systems must also be able to survive and operate in the radiation environment created by the detonation of nuclear weapons.
- High performance requirements based on the need for real time data signal processing and/or high data storage capability.

The failure or even transient mal-operation of a device in a critical circuit could result in either the loss of a satellite or prolonged unavailability, both of which could have severe economic or national security implications. Although the general scenario for an aircraft is less severe, microelectronics failures can also result in costly downtime and extreme passenger inconvenience with attendant deleterious economic impact.

Because of the extreme operating environment and the strict safety or mission requirements, it is easy to pigeonhole aerospace systems as an extreme exception to normal computational systems. As many aerospace organizations are often using the same or similar hardware as other consumers of commercial electronics, aerospace systems are a canary for impending reliability problems in other types of systems. Already, we have found similar problems in large-scale terrestrial applications, as the lessons learned from aerospace systems did not transition to supercomputer design. Without changes to system design, smaller safety-critical systems, such as medical and automotive technologies, will have more reliability issues as technologies scale. Therefore, the challenges that aerospace designers are facing currently are very likely an indicator of how terrestrial and safety-critical systems will be in the future if reliability problems are allowed to worsen as technology scales.

In the following sections of this report we will identify and discuss the most critical challenges imposed on aerospace microelectronics by the conflicting needs that arise when one is faced with the requirement to provide a system based on simultaneous SWaP, reliability, performance and cost constraints.

E.1.1 Background

While many organizations that build aerospace computing systems are not electronics manufacturers, they do significant amounts of system-level design, making aerospace organizations some of

the most expert consumers of electronics. Most organizations also undertake a great deal of the environmental testing, including radiation, mechanical, and thermal testing, necessary to ensure part and system reliability. Much of this data is being published openly in journals, such as the IEEE Transactions on Nuclear Science and IEEE Device and Materials Reliability. While there has been an increasing use of modeling tools to predict the radiation reliability of analog devices, these tools are not sophisticated enough to predict unexpected problems from bad design or manufacturing. For complex computational devices, such as field-programmable gate arrays (FPGAs), fault injection tools that simulate single-event upsets have been useful for predicting workload-specific behaviors, which has allowed organizations to use accelerated radiation experiments as a final verification tool instead of a discovery tool.

Since many organizations are trying to leverage as much commercial electronic technology as possible, the testing necessary to qualify devices for space and avionics programs puts them in a unique position to observe trends in both commercial and radiation-hardened technology. Modern SRAM devices often not only suffer from single-event upsets, but shrinking feature sizes have caused multiple-bit upsets to dominate [61, 44]. Both SRAM and DRAM memory devices are often plagued with either single-event latchup or high-current events that force the device to be frequently rebooted to avoid permanent device destruction, and single-event functional interrupts can cause widespread corruption of the memory array that stymie the most commonly used error correcting codes [71, 40, 57]. Many organizations are trying to use commercial computation devices, such as microprocessors and FPGAs, to increase the amount of computation that can be done in-situ before sending data off system [33]. Commercial devices have problems with single-event latchup and are particularly sensitive to accumulated dose effects, such as total ionizing dose and displacement damage [17, 16]. FPGAs can have a complex array of single-event functional interrupts, single-event transients and single-event upsets that cause device availability and data reliability issues [48, 10, 126]. As commercial computational components become more common in space systems, commercial analog devices for data movement have become more common. While commercial analog devices can provide faster data rates than radiation-hardened analog devices, commercial analog devices often experience single-event latchup in high radiation environments. Furthermore, many radiation problems are interlocked with power and thermal problems, where increases in temperature and decreases in voltage can cause a non-linear increase in radiation sensitivity [12].

In the next two sections we will discuss specific challenges for satellites and airplanes separately. For the remaining document, the discussion will summarize common challenge problems and potential solutions.

E.1.2 Satellite Overview

In comparison to traditional computing systems, satellite systems are relatively "flat" systems. Often, most of the computation is done in specialized hardware using a real-time operating system and very little software. Satellites can have complex hardware designs with relatively little use of software. As it is, many current satellites do not have enough software to make complex decisions and are commanded manually. On top of it, large satellites with multi-organization, multi-country collaborations often maintain isolation between separate sub-systems and limit manually commanding the individual instruments. In smaller satellites, much of the fault-protection system can be reduced to putting the system into a "safe state" by turning the satellite to the sun and waiting for a manual command from the ground station. While more autonomous satellites are desired, such satellites

would need to be able to make complex and reliable decisions on error-prone hardware. On one hand, automating some decisions will allow for quicker response in cases where the instruments might be damaged, such as overriding commands that could potentially damage instruments. On the other hand, there remain concerns that mal-functioning satellites can either potentially damage other spacecraft [1] or reveal sensitive information or hardware to other countries [4].

Future planned and envisioned spacecraft pose a number of unique challenges concerning microelectronics that can be delineated into two distinct categories:

- Traditional spacecraft are generally large and very costly systems that are expected to last for at least a decade on orbit. For example, this class of spacecraft can weigh more than 10,000 pounds, cost over $1B USD/satellite, take a decade to build, last longer than a decade on orbit, and be a multi-national collaboration with dozens of payloads. Satellites are deployed to accomplish specific missions, such as navigation (GPS), communications (Cable TV), intelligence gathering, or weather forecasting (NOAA).

- Micro or mini satellites are small satellites specifically designed for short duration missions. This class of spacecraft can weigh less than 100 pounds, cost $25M USD/satellite to build, take years to build, and last two to three years on orbit. Small satellites might only achieve one mission, such as a space weather experiment or a specific surveillance need. Small satellites might need to operate in a "swarm" of similar satellites to accomplish an overall mission.

The microelectronic challenges for each are somewhat different but equally daunting. Both are discussed in the following sections.

New Challenges for Traditional Spacecraft Emerging requirements for this class of spacecraft now call for extending mission life to greater than 15 years and flexible payload capabilities. In addition, to provide a better remote sensing capability many satellites will be required to either operate in a Medium Earth Orbit (MEO) at 2,000–25,000 kilometers above the earth or a prolonged South-Atlantic Anomaly (SAA) environment [130] which will impose the need for increased radiation hardening and fault tolerance.

Typical payload needs involve phased array antennas that may require significant on-board signal and data processing, which in the end will also drive a commensurate increase in data-storage capability; "smart" power management and distribution systems; far more capable command, control and data handling for the spacecraft; and built-in payload flexibility through hardware and software reconfiguration. In order to meet the processing needs, it will be necessary to employ an increasing number of advanced microelectronics devices, including > 500 MIPS microprocessors, solid state recorders, Gb SDRAMS, $>$ 14-bit ADCs, $>$ 1 Mgate SRAM-based FPGAs and other types of deep submicron (< 130 nm) devices with little or no space heritage and that are not, in general, designed for either high reliability or radiation exposure. Based on this increasing dependence on commercial technology, a "layered" approach to fault mitigation and tolerance will be needed for large-scale satellite systems. Already, there are some point solutions that use multiple devices to increase reliability, such as memory or FPGA scrubbing devices [20] that remove errors from memory-based devices. For FPGAs, the scrubbing circuit plays the role of a watchdog that monitors the state of the health of the other device. In addition, improved and cost-effective testing, screening and qualification approaches will be needed.

Thus, the basic challenges for the advanced technology microelectronics for this class of system are:

- Reducing the cost of testing and qualifying components and instruments for space
- Allowing for more systematic and complete exploration of reliability, robustness, and performance while reducing design time and costs
- Removing the isolation between design layers that prevents opportunities for synergistic, multi-layer system optimization, resulting in reduced capabilities in a given size/weight/power/reliability envelope

Micro/Mini Satellite Challenges While many of the above noted challenges apply, there are a few added ones imposed by this class of spacecraft. The challenges for small satellites include, but are not limited to:

- The need for increasingly robust microelectronics based on the added vulnerability from the use of composite materials for the satellite structure that are thinner and lighter than previous materials used. As these materials provide less shielding from radiation, devices will experience an increase of radiation effects from single-event effects and the accumulation of total dose.
- Multi-dimensional design tools that can address satellite "swarm" availability and performance requirements to simultaneously optimize performance, number of assets required, and individual asset requirements/capability.

Space Discussion Based on the above discussion concerning satellite system needs, we can provide a listing of cross-layer challenges that must be addressed to support future aerospace microelectronics and electronic system design. The satellite-specific challenges include, but are not limited to:

- Reducing the cost of testing and qualifying components and instruments for space
- Allowing for more systematic and complete exploration of reliability, robustness, performance, weight, and survivability while reducing design time and costs.
- Allowing for reliable circuit and sub-system designs using state-of-the-art hardware, including multicore processors, so that mission requirements for reliability and radiation robustness can be met with modern hardware devices.
- Calculating overall system fault grading and composable error rates/failure estimates for multi-component systems.

Thus, it is recommended that a program that addresses cross-layer challenges be initiated.

E.1.3 Airplane Overview

Airplane systems have similar challenges to satellite systems, because many airborne applications use commercial electronics in harsher environments with longer life requirements than typical ground applications. Airplane systems contrast to space systems in that their usage environments may be less severe in terms of thermal extremes or radiation exposure, but more severe in terms of

vibration and thermal cycling. Airplanes may have several cycles (takeoff, cruise, landing) daily and operate in a variety of locations with significantly different thermal, moisture and contaminant exposures. Unlike space systems, aircraft also have comparatively easy and frequent access to maintenance.

Reliability requirements for airborne electronic systems vary depending on the criticality of application, often characterized by three major levels: flight critical, mission critical, and non-critical. Flight critical applications are those involved with controlling flight, have the highest level of reliability and safety requirements, and often utilize older, established legacy electronics for which reliability can be assured. Mission critical applications in military aircraft are those involved with mission planning, identification and response to threats, situation awareness, and communication with other platforms. For airplane systems, performance and reliability are both important, but newer electronics are desired to provide faster, more accurate, and more detailed solutions. Multilevel reliability and fault tolerance are important considerations for mission critical applications. Non-critical applications are any other type of electronics, such as those providing office functions or passenger entertainment.

Although aircraft can have a platform life requirement of 30 years or more, the electronics on board may be replaced more often, and may have planned technology refresh cycles of 5 to 10 years.

New Challenges for Aircraft Applications There is a growing dependence on Commercial-Off-The-Shelf (COTS) electronics in military and commercial aircraft platforms to provide unique, faster, and more competitive functional capabilities while minimizing total ownership costs. As feature sizes decrease, electronics become more susceptible to airborne environments, especially atmospheric radiation. Newer technologies (< 100 nm) may have shorter expected life than required or needed, due to increased susceptibility to environmental fatigue factors.

Some specific airplane challenge areas:

- Environmental qualification and testing to provide reliability of commercial electronics in airborne environments.
- Security and prevention of unauthorized access or sabotage. This challenge may drive additional fault-tolerance measures incorporated in a multi-layer system design.
- Lightweight solutions that tolerate increasing aging effects and in-system device failures. Examples might include multi-layer data integrity assurance features, such as health monitoring, troubleshooting, and health management features. Reliability sensors and/or circuitry designed to indicate exposure levels to harsh environments (e.g. "cage canary" circuits) could help in this.
- Efficiently addressing a wide range of reliability requirements. This might suggest tunable reliability, where the ability to tune in high-reliability features or not would allow the airplane or the operator to change the reliability of the system depending on mission needs at different times (e.g. ability to go to "red alert").

Discussion of Challenge Problems There are a number of challenge problems identified by the Aerospace working group:

- Widening Gap Between Mil/Aero and Commercial Parts

- Design for Worst-Case Environment
- Multidimensional Optimization Problem
- Testing Bottleneck
- Part vs. System Reliability
- Flexible Mission/Science vs. Fixed Capabilities
- Assuring Supply Chain
- Increased Aging Effects

An overview of the aerospace challenge problems is provided below.

Widening Gap Between Mil/Aero and Commercial Parts: Over the years, radiation-hardened space technologies have lagged commercial technologies by as much as 10–20 years. Despite the fact that existing defense and space semiconductor markets are becoming increasingly nonviable, a recent push for "trusted foundry" [77] electronics is forcing designers to use Mil/Aero devices. Leveraging commercial technologies would allow designers to employ more aggressive components in aerospace systems, but would require part- and system-mitigation techniques.

Design for Worst-Case Environment: Many satellites are expected to endure years in a harsh radiation environments with rapid temperature cycling and an initial intense vibration. Airplanes are also designed for a similar environment with more mechanical problems caused by repeated turbulence and landing conditions. As many designers are margining for worst-cases in thermal, radiation, and mechanical situations, it is possible that designers will margin for the scenario when all three areas are in problematic scenarios at once, which could lead to extreme worst-case margining for a scenario that is unlikely to occur.

Multidimensional Optimization Problem: Most satellite/avionics designers are trying to optimize performance, reliability, power, thermal and weight. Mission requirements often impose fixed hard limits on power, thermal, reliability and weight. Optimizing these limits can be very hard to manage as changes along one dimension (power, thermal, reliability, weight) affect the other dimensions. Furthermore, the power, thermal, reliability, and weight problems are often the responsibility of different teams with different skill sets. As no tool currently exists to optimize all of the problems at one time, it is currently done manually. While experienced systems-design teams often come to very good solutions, the tendency is to over-design the system.

While there is a low power density for space, aircraft have a far less constrained power supply. Unfortunately, more powerful systems often translate to heavier systems and heavier systems cost more money to fly. Therefore, minimizing weight for airplanes is necessary.

Testing Bottleneck: Many organizations are responsible for initial environmental testing to determine worst-case reliability calculations in an attempt to eliminate parts that will not meet mission requirements or will be too difficult to work with. The advantage of using Mil/Aero or "heritage" parts (i.e., devices used in previous space missions) is that initial testing can be eliminated or minimized. The disadvantage of using commercial parts is that often times the organization will need to do all of the initial environmental testing, which can be costly. In recent years, dynamic testing to determine workload-specific reliability has become necessary. For some devices, such as FPGAs, fault injection can be helpful in minimizing the testing burden for dynamic testing. Because inadequate/incomplete testing can lead to false security and in-field system failures, most organizations will pay to get it right, which leads to conservative over-design or over-testing of the system.

Part vs. System Reliability: System reliability is often considered intractable, so many organizations focus on part reliability. While part-specific error mitigation methods can be useful

and should be encouraged, there are only a few point solutions available for solving reliability problems at the system level. The side effect of this situation is that it encourages wide "margin of error" design practices that can lead to slow, heavy, power-hungry, and expensive solutions.

Flexible Mission/Science vs. Fixed Capabilities: Currently, the science and national security concerns that drive our space programs change several times over the course of a satellite's lifetime. Most satellites are commissioned and designed for a specific science and/or national mission need and the only access to post-deployment changes is through software. On top of this, satellites can be damaged during and after deployment, making them unusable. Due to the growing need for "opportunistic" missions, many satellite operators do attempt to make "heroic" changes to satellites to collect necessary data, which can irreparably damage a satellite. The only current solution is to launch new satellites, which could cost billions and take years to build. More adaptive satellite design could weather changes in the science, the mission, and the satellite while deployed.

There is a similar need in airplanes to be more flexible. Currently, even minor modifications to the software of an airplane can force the airplane to be re-certified. As re-certification can take months, this situation deprives designers of the ability to use late-bound information to change the system for the better.

Assuring Supply Chain: For military and national security missions, guaranteeing that parts are not counterfeit or have not been tampered with has become difficult. This problem has lead to heavy reliance on trusted foundry hardware.

Increased Aging Effects Newer technologies (< 100 nm) may have expected lifetimes that are too short to be economically accommodated, demanding frequent manual repair and/or excessive over-provisioning of resources. These shorter lifetimes arise from increasing susceptibility to environmental fatigue factors and aging mechanisms [37]. Without new mitigation techniques, this could prevent access to advanced technologies.

The above set of problems leads to a culture of conservative over-design of aerospace systems. The aerospace community is often forced to avoid deploying new electronics that could enable better in-situ processing and/or decision making. Currently, our satellites are becoming rapidly overburdened. Over the past decade, as our threats have increased, the need for more satellite coverage and more data collection has out-stripped our satellites' capacities. At the same time, other countries have found access to space easier, as these countries are not as concerned with mission failure. Even though we are currently technologically superior to nearly every other nation in space, this gap could close rapidly. Conservative, manual, over-design to meet strict mission requirements will not help us remain technologically superior. Remaining technically aggressive by leveraging new computational technology with shorter lifetime and experimental spacecraft will help us not only maintain our technological superiority but can increase national security at the same time. Doing this responsibly demands that we find suitable system-level techniques to mitigate the reliability problems of modern, commercial technologies so that system reliability goals are not compromised in the process.

E.1.4 Potential Solutions

The working group proposed and discussed a number of solutions to the challenges described above. These discussions are detailed below for reference, but they should be no way regarded as a complete discussion of potential approaches.

Modeling the (Performance, Reliability, Power, Thermal) Tradespace: By encapsulating

information from the devices, designers can work at the system level, instead of at the part level. To be able to help designers, modeling tools that manage the complexity of these problems are needed:

- Memory architecture solutions for known device sensitivities
- System reliability analysis
- (Reliability, Performance, Power, Temperature) optimization
- Methods for managing multiple reliability problems
- Simulation of lifetime aging problems
- "Day in the Life" simulations

Agile Satellite Solutions: By using adaptive or multi-level reliability solutions, satellite capabilities would not be as static. More adaptable satellites would be able to change reliability requirements over different phases of the mission life. By linking satellites to radiation monitoring systems (either in the loop or data updates from ground stations), the spacecraft should be able to adapt to changing space weather conditions once a threshold for corrective action is met. Furthermore, multi-level reliability solutions that use either software-based hardware checking for fault tolerance, such as the Hubble servicing mission, or reconfiguration to adapt to space weather, lifetime aging, and science changes will create more flexible satellites. Finally, to make this type of satellite work, autonomous analysis tools and methods for assuring continued functionality are needed.

Multi-Core Solutions that Address a Wide Application Space: Currently, many aerospace organizations are catching up to multi-core computing. At this stage, there is very little understanding about how multicore devices will work in high-reliability environments. As long as common failure modes are not likely, many possible mitigation strategies will exist, but there is currently not enough information about multicore-specific failure modes.

Dual Purpose Reliable and Secure Computing Solutions: Many of the same ideas that researchers have been working on for reliability would also be good for security. In particular, fault encapsulation could be useful for both reliability and security by keeping intentional or random errors from hurting the data or the system. Tools that can handle both problems are needed as well. Combined reliability/security tools will allow designers to achieve both secure and reliable designs, will provide test methodologies (modeling, fault injection, field tests) to validate systems, and will provide system tests for in-field operation.

E.1.5 Conclusions

In this paper, we have presented a number of challenges that face aerospace designers. Due to the strict constraints on aerospace systems and the harsh operating environments, designers are often trying to find the best solution to multiple reliability problems. The use of cross-layer reliability solutions would allow designers to build more flexible and agile systems that would allow them to adapt to the many challenges ahead for aerospace systems.

E.2 Consumer Electronics

Authors: *Nicholas P. Carter (Intel), Todd Austin (UMich), Kimmo Kusilinna (Nokia), Chris Wilkerson (Intel), Andrew Huang (Chumby), and Helia Naeimi (Intel)*

"Consumer Electronics" are defined here as computing/electronic systems that don't fit into one of the categories "life-critical systems," "aerospace systems," "infrastructure," or "large-scale." The category is, obviously, vague. In particular, it's hard to draw an exact line between the largest "consumer" computing system and the smallest "large-scale" system.

Over the last several decades, the exponential rate of improvement in both transistor performance and device density known as Moore's Law has been widely lauded as the reason for the growth and success of the consumer electronics and computing industry. However, this interpretation is not quite correct, in that it neglects the fact that users do not see fabrication improvements directly, but instead see improvements in the performance and capabilities of the systems that engineers are able to design using improved fabrication technology. It is these improvements in system performance and capabilities that drive new applications, new products, and users' desire to upgrade their systems well before they physically fail.

This distinction may seem minor, but it motivates the key challenge facing computer electronics/computing over the next 10–20 years: the need to develop designs that tolerate increasing rates of device-level variation, unpredictability, and failures without devoting so much silicon area and power to tolerating these effects that the rate of improvement in user-visible performance over time decreases significantly. Put in a more qualitative fashion, each new "generation" of fabrication technology approximately doubles the number of transistors that can be built on a chip of a given size, but this increase in device density comes at the cost of increased variation and error rates. If too many of the new transistors that a given fabrication process provides must be devoted to tolerating the process' increases in variation and error rates, products built in that fabrication process will not deliver enough improvement in user-level performance or capabilities as compared to products built in the previous fabrication process to motivate consumers to purchase them. If this happens, the growth engine of the electronics/computing industry will stall, because the industry relies on the profits from products implemented in each fabrication technology to fund the development of the next generation.

While errors in computation have been an issue for consumer systems since at least the 1970s, when researchers began to study the rates of alpha particle-induced soft errors in DRAMs, two trends argue that designers will need to shift from the current model of error correction in consumer electronics, which applies individual correction mechanisms to the structures that see the highest error rates, to a more system-level model in which the entire system considers the possibility of errors and variation. First, increasing rates of errors and variation are making it increasingly difficult to deliver sufficient reliability through a collection of mechanisms that tolerate individual causes of errors, such as ECC bits on memory arrays, increasing both the number of mechanisms required by a design and the hardware cost of each mechanism.

Second, designs are increasingly becoming limited by a chip's power budget instead of by the number of transistors that can be fabricated in a given amount of chip area, motivating the desire to reduce the "guard bands" on a chip's power supply and clock rate. Current designs operate at power supply/clock rate combinations that are significantly lower than their peak capabilities in order to ensure that they will continue to have very low error rates even when operated under worst-case conditions and/or at the end of their product lifetimes. In contrast, designs that are able to detect and correct timing errors are often able to operate at significantly (30%) more efficient power supply/clock rate combinations when implemented in conventional CMOS. This power/performance benefit from reducing guard bands is expected to increase as fabrication processes scale, due to increasing device variation and sensitivity. Similarly, the efficiency advantages of error-tolerant

designs become even higher when implemented in near-threshold-voltage CMOS, because of the higher performance variations seen in such designs.

Because of these and other constraints, we believe that consumer electronic/computing systems will need to adopt full-system approaches to reliability, in which multiple levels of the system stack collaborate to detect, tolerate, and adapt to errors and variation. Developing these approaches will require research at all levels of the system stack and vastly-increased communication and collaboration between researchers at different levels in the stack. To facilitate, guide, and support this research, we have identified the following high-level research focus areas for reliable consumer computing:

1. **Models and abstractions for errors and variation:** Current research tends to focus on solutions to specific physical causes of errors and variation (SEUs, NBTI, etc.), leading to a profusion of extremely-specialized techniques. Developing a small set of abstract categories of errors/variations and showing that a wide range of physical effects can be coerced into one or more of those categories would simplify system design and analysis. Further, it would encourage and support the creation of clean communication interfaces between layers in the system stack by abstracting away some of the less-important details of the physical causes of errors and variation.

2. **A general framework for multi-level reliability/resilience:** One of the difficulties facing researchers is the lack of a "standard" architecture/system stack for a resilient system into which they can insert new techniques. Similar to the way the availability of a "conventional" model of superscalar architecture and tools to model such architectures allowed computer architects to develop new micro-architectural mechanisms without having to re-implement entire processor designs, the availability of a framework for reliable systems and one or more exemplar designs would make it possible for researchers to focus on individual aspects of reliable system design and to compare their techniques to other approaches with some confidence that they are doing a fair comparison. Having such a framework/toolset would also increase smaller institutions' ability to contribute to reliability research, by lowering the "barrier of entry" required to generate useful results.

3. **Testing/verification strategies for reliable systems:** Systems that tolerate/correct errors promise to increase fabrication yields by providing correct operation in the face of a small number of fabrication defects, but their very resilience makes it difficult to test them at fabrication time by increasing the number of logic paths that must be tested and by hiding defects. As resilient designs become commonplace, it will no longer be sufficient to merely determine whether or not a chip is functionally correct at fabrication time. Instead, it will be necessary to characterize both functional correctness and the amount of "safety margin" remaining in terms of the chip's ability to tolerate in-field failures and errors before a chip is declared ready to ship, and it will be necessary to do so without significantly increasing test time and cost, which is already a significant issue in the electronics industry. Similarly, it will be necessary to develop in-field diagnostics and testing techniques that can monitor a system's state as it ages in order to tolerate changes in circuit behavior and predict when a chip's reliability will drop below the requirements of the system due to accumulated errors and variation.

4. **Improved recovery/rollback mechanisms:** While this is a subset of the general reliability problem, recovery and rollback in consumer-scale systems is one of the least-studied aspects

of the reliability space, and will need significant attention in order to avoid spending excessive circuitry and power handling infrequent events. In particular, it is likely that future systems will incorporate a hierarchy of recovery mechanisms with different costs and capabilities, such as pipeline squashing, checkpointing at different levels in the memory hierarchy, and infrequent checkpointing to non-volatile storage.

5. **Lightweight detection:** The costs of recovery and rollback are closely tied to the latency of a system's error detection mechanisms. Low-latency error detection significantly reduces the cost of recovering from errors by reducing the amount of work that must be "undone" to restore the system to a state before the error occurred. To maximize efficiency, future systems will require a variety of error-detection mechanisms that are optimized to minimize both error detection latency and overhead. These mechanisms should work with the recovery/rollback mechanism by not only determining what has happened when an error occurs but by also bounding the amount of time that has passed since the error occurred, allowing the system to select a recovery/rollback mechanism that minimizes the cost of recovering from the error.

6. **Interfaces and abstractions for reliable system-on-chip design:** While microprocessor architectures receive a great deal of attention, more and more product designs are being done using a system-on-chip approach, and this trend is expected to continue as improvements in device density allow larger portions of a system to be integrated onto a single chip. In particular, systems designed by connecting multiple pre-designed circuit blocks through standardized interface architectures and a small amount of custom logic are becoming extremely common. As it becomes important to consider reliability at all points in the product spectrum, it will become necessary to develop standardized interfaces for reliability, error notification, retry, and reconfiguration in SOC designs.

7. **Scalable approaches to and abstractions for reliability:** Consumer electronics are extremely sensitive to the costs of providing reliability because they compete in a marketplace in which performance (or performance per unit power or cost) is critical and errors are relatively rare. In contrast, other portions of the industry (aerospace, HPC, etc.) have higher error rates and greater error impact (a crashed airplane as compared to the need to reboot a laptop), but are less-sensitive to cost and overhead. Scalable reliability techniques, which allow system integrators to trade off overhead against system reliability, might make it possible to use the same designs in both consumer and high-reliability products, allowing the high-reliability portions of the industry to benefit from the sales volumes of the consumer electronics industry.

E.3 Infrastructure

Authors: *Zbigniew Kalbarczyk (UIUC), Allan Silburt (CISCO), Shi-Jie Wen (CISCO), Karthik Pattabiraman (Microsoft), André DeHon (Penn)*

Our modern infrastructure is highly computerized. This infrastructure includes our power grid, our building control (heating and cooling, fire suppression, security), and our telecommunications (phone, internet, cable). All of these are directly or indirectly life-critical: communication is necessary for emergency response, power is required to run emergency and hospital equipment as well as to keep our environments livable, and much of building control is associated with keeping us safe.

Because of the additional capabilities and economics they provide, the trend over time has been to increase the computational components of these systems. Automation responds more quickly and consistently than humans and makes our systems run more efficiently. All this means we must provide high reliability for an ever growing computerized system.

Because of our increasing dependence on computational and communication infrastructure (networking, computing, cloud computing—the combination of the network and the compute servers), outages also have a large, negative economic impact. Many modern workplaces grind to a halt when the network is out, resulting in large costs (*e.g.*, consider the professional salaries of the impacted populace for the period of the outage. Alternately, consider the lost sales and reputation due to the outage). In cases where computation controls a larger physical plant (*e.g.*, power, heating, cooling), failures of the computation to provide appropriate control could endanger the controlled plan (*e.g.* allow a power line to overload or a chemical reaction to proceed out of control).

Infrastructure systems tend to be highly distributed. In many cases, their spatial distribution is essential to the services they provide—we must get power out to a large area, communication is about connecting distant people and machines, and building control must reach into all spaces in a building. This means computations cannot be centralized in a carefully controlled environment and are less physically accessible. It also means that system upgrades do not occur uniformly and the system as a whole will almost always be composed of many different generations of technology.

Computation in some of these infrastructure roles (*e.g.* power, heating, cooling) is relatively inexpensive compared to the plant the computation is monitoring or controlling. As a result, this class of system has been able to tolerate larger overhead costs for reliability (*e.g.* if the computing is only 1% of the cost of the system, duplicating or triplicating it may only increase the system costs by 1–2%).

Availability is a key metric. What is the fraction of down time? Short service failures (few milliseconds of network outage, few seconds of heating or cooling control) may be tolerable, so infrastructure systems care both about the frequency of upsets and the time to recover. Long outage events must be very infrequent, whereas quick recovery events can occur at higher frequencies.

Increasing efficiency demands greater computational control, either to control more things or to find solutions closer to the optimum. This increases computational needs, but not excessively. Many Green initiatives to reduce energy consumption rely on more sophisticated computation and monitoring to control energy usage.

Much of the economics of the computing infrastructure (perhaps more so in the context of networking and telecommunications) come from riding the main-stream technology wave. So, while computing needs in some infrastructural areas (perhaps power and building control) might be satisfied with a freeze at 180nm technology, this now places a premium cost on maintaining access to older technology, one that will make the electronic parts even more expensive. The coupling and volume benefits between industries remains strong.

Challenges

- Affordably increase availability for increasingly large and distributed infrastructure systems.
  - Availability cannot decrease.
  - With increasing system sizes, this suggests an increasing reliability demand on each component.
  - While computation is often not the dominant costs in these systems, the cost for the computation cannot increase significantly.

  - Like life critical and aerospace systems, the availability and reliability demands for infrastructure is often higher than consumer components. At the same time, these applications cannot afford completely unique components for their use. This suggests these systems may benefit from adaptable systems where they can use standard components and configure them to provide higher reliability levels when serving in this role.

- Infrastructure cannot afford to develop all components custom for their applications and systems or to maintain processes and technologies unique from other market segments.
  - This situation motivates the design of components and systems with modes and configuration options that allow higher layers in the system to tune what the components spend on reliability based on their market segment.
- While compute costs do not currently dominate, it, nonetheless, remains true that providing complete, guaranteed never-fail service is prohibitively expensive. This suggests a need to increase availability by:
  - providing degraded modes of operation that continue to provide some level of availability when failures occur.
  - providing affordable monitoring of the distributed infrastructure to allow early warnings of problems and to rapidly diagnose failures and expedite repairs. Affordability in some cases will mean extremely low power for the monitors, suggesting a sensitivity to monitoring power requirements.
  - supporting remote reconfiguration and repair to rapidly restore some level of service.
- Human service costs must decrease despite increasing component complexity, increasing system size, and increasing distribution of components. Advanced technologies that may see earlier wear-out exacerbate this challenge.
  - This suggests a greater need for automated and remote repair and adaptation. It also suggests a need for adaptation to integrate, accommodate, and optimize systems composed of many heterogeneous technology generations.
- Analog sensors, discrete components, and passive are not reliable enough.
  - This suggests the need for mitigation at higher levels in the system stack for truly reliable solutions, which, in turn, suggests a need for standard interfaces to increase the observability, diagnosability, and control of the sensors.

E.4 Life-Critical Systems

Authors: *Mark Porter (Medtronic), Glenn Forman (General Electric), Kevin Kemp (Freescale), Claude Moughanni (Freescale), Tony Reipold (Freescale), Xiaowei Zhu (Texas Instruments)*

As electronic systems become more pervasive in society, the complexity of the hardware and software used in applications where malfunctions could cause serious injury or death is also growing. The desire to construct such systems stems from the significant benefits they provide. Implantable pacemakers and defibrillators provide life-support to people with heart conditions that might otherwise prove to be fatal; airbag deployment and traction-stability control in automobiles improve the chance of survival in dangerous accident situations; beam control and placement in

radiotherapy suites allow doctors to destroy malignant tumors while minimizing damage to healthy tissue; diagnostic imagery enables the identification and treatment of diseases and injuries and significantly increases beneficial medical outcomes; air transportation is optimized for millions of passengers a day through advanced air traffic control.

Reliability research has become more challenging due to increased system complexity, reduced development timelines, smaller feature sizes, and the demand for new product availability. Accelerated testing can fail to match real-world experience due to component and system miniaturization, and may lead to inferior reliability models. More predictive methods and more resilient architectures are needed.

Life/safety-critical systems that employ next-generation computing devices will require increasingly productive and accessible reliability tools, models, and data. In addition, deep-submicron CMOS scaling is driving the need for a more advanced understanding of emerging pathogenic IC mechanisms. Reliable computing in the presence of these failure mechanisms will require an affordable, comprehensive immunization against fault propagation with the use of novel cross-layer architectural solutions.

Resilient systems for safety-critical applications must incorporate the benefit of cross-layer improvements for next-generation and future-generation electronic IC components, as well as packaging and interconnects to achieve total system solutions. In many instances, safety-critical designs tend to lag behind the state of the art in IC technology. However, more advanced medical life sustaining, prosthetic limb/vision, and molecular imaging equipment requires the best and highest-performance computing available.

This research aims to change the nature of the host response to random and systemic faults, and to better understand the pathogenic role of known and emerging causes. This includes increasing transistor variation due to scaling, soft-error vulnerability, and permanent faults due to device wear-out.

In aerospace and defense, certain device designs have addressed upsets by such methods as hardened-by-design, redundancy, or hot-spares. However, the increased power, area, and cost of such brute force defenses are counter to economic, commercial, and performance forces that represent the computing industry's more substantial and fundamental market drivers. Such solutions are also less suitable to implantable devices and portable/wearable medical instruments due to their larger size, weight, and power. Ultimately, our effort to address pathogens by newly formed methods of cross-layer immunization will be a balance between costs and benefits. By using cross-layer mechanisms, more affordable and relevant life/safety-critical solutions will ultimately be discovered.

E.4.1 Implantable Medical Devices

The first implantable pacemaker was developed in the late 1950s using solid-state transistors that allowed the devices to be small enough to run on battery power. Over the decades since that time, improvements in available technology to design and build these devices have allowed an ever-wider array of therapies to be delivered to patients to treat many additional forms of illness.

The medical device industry has used a deliberate strategy of lagging the leading edge, especially of semiconductor technology. As transistor dimensions have shrunk along the Moore's Law curve, medical devices continue to use geometries that are several generations behind the high performance computer industry. We expect this conservative design principal to continue for two main

reasons:

1. Leading edge technologies undergo a yield learning curve that requires a period of time to elapse before defect density has been substantially reduced. Reliability also suffers during this early period due to a higher number of latent defects. As transistor geometries have shrunk to below 0.25μm, off-state leakage currents have also grown. In sub-65nm technologies, gate leakage has become such a significant problem for commercial applications that new materials have been introduced to counter the effect. In the medical device industry, where battery lifetimes are required to meet 10-year longevity, the current drain of advanced CMOS technologies is unsustainable.

2. Countering this trend is the desire to provide both higher reliability and additional performance. As the complexity of closed-loop feedback systems and diagnostic data sets grows, there is a need for advanced technologies to provide a boost in performance without a corresponding increase in energy usage. As part of the solution to address this application space, the need to ensure high reliability among all the components of the system will require new paradigms in design, test, and resiliency.

E.4.2 Automotive Electronics

The last several decades have seen explosive growth in the application of semiconductor electronics in automobiles, already exceeding 70 microcontrollers on some high-end vehicles. Some of these include: engine and power train control to maximize performance and fuel efficiency; driver and passenger information, comfort and entertainment systems; active safety systems such as airbag deployment and seat belt pre-tension devices; and vehicle dynamic safety and control including anti-lock brakes, traction control and electronic stability programs. Newer safety-oriented applications include adaptive driver assistance systems using radar, active cruise control, lane departure warning and night-vision systems, while future concepts may include electric drive-by-wire steering and braking. The ultimate paradigm for automotive transportation is that of fully autonomous vehicles operating on an intelligent highway with no need for a human driver at all.

Driven by consumer quality expectations, warranty service and recall costs, and their increased use in safety/life-critical applications, automotive electronics are required to meet unprecedented levels of reliability and resiliency against potential failure mechanisms. In addition, these systems are required to operate for 10-20 years in harsh environments including extreme thermal, moisture, vibration, and electromagnetic noise conditions.

E.4.3 Challenges

Critical to the continued performance and reliability improvement of safety-critical systems is the ability to abstract the defects and errors that can occur during design and manufacturing into a set of rules that can be used to guide resilient architecture choices.

1. Fault-tolerance/resiliency design choices are difficult to implement in a timely and efficient manner. This reduces both the number and robustness of the candidates and limits tradeoff options. Only the simplest or most critical cases are generally addressed.

2. Error correlation between device physics and architectural effect is poorly characterized and categorized. It is extremely difficult for designers to understand how any particular failure, hard or transient, will expose weaknesses in a given design.

3. Sensors, discrete components, and passives are essential elements for most safety-critical systems and can limit overall system reliability, but are generally ignored in the literature of resilient design. Safety-critical systems must have visibility into defects of all components, not just CMOS, and be prepared to address their failure or misbehavior.

4. It is often unclear how or where to make tradeoffs between the electronic components of a safety-critical system and the application in which it is embedded, where other resiliency techniques may be available. For example, in an automotive environment there exist mechanical back-up systems that can take over for malfunctioning electronics.

5. Translation from concepts to practice is hindered by the lack of a standard model for describing, comparing, and composing resilience techniques. Single-layer resiliency techniques provide complete solutions only in very limited circumstances (*e.g.* memory ECC); multi-layer protocols built in an ad-hoc fashion are expensive and non-portable between applications or system architectures. Comparing design solutions, especially as proposed by multiple authors in the literature, is difficult or impossible without a complete understanding of all the nuances of every aspect of the design, rendering the information less useful to new architectural designs.

6. Exploring and characterizing resilient/fault-tolerant design techniques and their efficacy in simulation and/or hardware is time-consuming and expensive.

   In addition to the need for research into the design resiliency challenges listed above, safety-critical systems will increasingly be required to conform to standards definitions that govern their design, performance validation, and maintenance (*e.g.* IEC 61508 or ISO26262).

7. It is currently difficult or impractical to integrate safety standard protocols and certifications into design flows. This challenge places a significant cost and effort burden on manufacturers that prevents many companies from adhering to standards that could increase consumer safety. This suggests the need to correlate standards with quantifiable increases in safety and to better align tools, design flows, and automation with standard specifications.

E.5 Large-Scale Systems

Authors: *Sarah Michalak (LANL), Dennis Abts (Google), Nathan DeBardeleben (LANL), Greg Bronevetsky (LLNL), John Daly (DoD), Armando Fox (UCB), Jon Stearley (SNL), David Walker (Princeton), Ravi Iyer (UIUC), Will Jones (Coastal Carolina Univ.)*

Large-scale systems are at a crossroads. As the feature sizes of electronics grow smaller and large systems grow to thousands of nodes and millions of cores, the probability that some system component will fail grows steadily. Today's systems have already begun to suffer from this trend, with many systems featuring more than one failure per day, including both fail-stop failures and data corruptions (ex: a 100,000-node BlueGene/L supercomputer suffers from one data corruption every

3–4 hours). This has led many applications and systems to explicitly incorporate fault tolerance features into their code, trading off the cost of fault tolerance mechanisms such as checkpointing against the cost of failures. A major example is the decision by major system vendors to remove local disks from compute nodes, which improves hardware reliability by removing the least reliable system component while making checkpointing significantly more expensive (~30 minutes on a typical large system).

Looking into the future, as large-scale systems grow even more complex and include increasing component counts, we anticipate a crisis of reliability where the underlying hardware will become too unreliable to provide useful service. As such, managing reliability at the level of individual components, entire systems, and applications must become a key pillar of large-scale system research and development. Since traditional reliability research has primarily focused on preserving the abstraction of perfect reliability in low-fault environments, this new problem of operating in the presence of many faults presents a new generation of system reliability research challenges that requires a significant new research investment.

At its core, a given fault or fault tolerance strategy can affect an application in one of three ways:

- Time: the application may run more slowly (*e.g.* the fault itself or the algorithm to tolerate the fault causes a degradation in system performance)
- Energy: the application uses more energy to produce its result (*e.g.* checkpointing requires some work to be re-executed and modular redundancy runs an identical application on multiple processors)
- Correctness: the application produces results that have a lower accuracy (*e.g.* a memory bit-flip may affect a numerical algorithm's convergence properties)

As such, as part of an integrated fault tolerance strategy we must (i) provide accurate characterization of how an application or system performs with respect to multiple metrics and (ii) develop ways to both improve performance and to allow applications and systems to trade off one metric, such as correctness, for another, such as completion time.

Challenge 1: *Understand and control the complex effects of faults on systems*

Large-scale systems consist of thousands or even millions of software and hardware components that interact in complex ways. Faults in one component can propagate through other components in many different ways to manifest themselves as a variety of application and system errors. The effects of a given fault may be different depending on the context and one part of the same application or system may be more or less vulnerable to faults than another. As such, system reliability solutions that treat all faults as equally dangerous and all components as equally vulnerable will be highly sub-optimal, giving up significant performance and functionality. Further, without the knowledge of how individual system components are affected by faults and how those faults travel through the system, it becomes very difficult to handle faults or to identify their root causes. Unfortunately, while today there exist techniques to understand the fault properties of individual software and hardware components, there exists little work on understanding the effect of faults on entire systems. As we work towards developing future generations of cost-effective and productive large-scale systems it is thus critical to overcome this limitation.

A simple example of how our lack of understanding hinders our ability to build reliable systems is the effect of bit flips on applications running on the BlueGene/L supercomputer, where a 100,000

node machine suffers from one L1 cache bit flip every 3-4 hours. Although these flips are detected using a parity code, they can only be corrected by running the L1 cache in write-through mode, which guarantees that there is a valid copy of the cache line in the L2 cache. Without knowing anything about the vulnerability of applications to bit flips, the best strategy is to assume that every single bit-flip is fatal to the application. As such, BlueGene/L designers chose to handle all such faults at the hardware and kernel level by providing users with two options: (i) run using L1 write-through mode, which can reduce performance by as much as 50% or (ii) abort the job when a bit-flip is detected, which requires expensive checkpoint/restart operations (on this machine a full-system checkpoint takes approximately 30 minutes). However, these costs can be significantly reduced if developers understand their application's true vulnerability properties. For example, Monte-Carlo simulations are insensitive to rare data corruptions, meaning that they require no specialized reaction to bit flips. Other applications may be sensitive but may develop efficient strategies for detecting and correcting bit flips if they are informed of them. This was the case for the ddcMD code, where performance was improved by 17% via the use of light-weight checkpointing to correct detected bit flips with no application aborts. Overall, the best strategy to employ on a given physical fault varies widely depending on the fault vulnerability of other system components and requires the cooperation of multiple components, each of which performs the fault detection/correction task that it is best suited for.

This means that cost-effective reliable computing requires a detailed understanding of the effects of component faults on entire systems and cross-component cooperation.

Sub-Goal 1a: Effect of faults on systems - Develop a detailed understanding of how faults propagate through large systems and manifest themselves as errors in other components. Although component-level reliability has been studied extensively in the context of individual devices, we do not currently have a good understanding of how failures of such components affect other portions of the system or the applications that run on top of it. The lack of such an understanding makes it difficult to predict the portions of systems and applications most vulnerable to failures or to identify the failed system components based on the effects of these failures on the system as a whole.

This is critically important because most work in "classical" fault tolerance focuses on *completely masking* faults from higher layers. As companies such as Google have discovered, at extreme scale this approach is not feasible; we must recognize that *not every fault has equally severe consequences*, and focus our efforts on how to selectively mask the most dire faults while potentially allowing higher layers of the system to deal explicitly with other faults. To do this, we will need to develop composable models of system vulnerability to failures that can predict how errors travel through and affect system components and applications. These models would allow us to:

- Design applications and systems to detect and tolerate the most likely or dangerous types of failures, including the degree of reliability truly needed from various components
- Develop tools to quickly identify faulty components, enabling efficient system management
- Predict many types of failure ahead of time to allow proactive fault tolerance strategies
- Apply different masking and recovery strategies to different faults depending on their overall impact on job completion.

Sub-Goal 1b: Cross-Layer Reliability - Develop tools and frameworks to enable individual components to participate in a global fault reliability strategy by defining ways for them to interact and share reliability information.

This problem of sharing the responsibility of system reliability across system layers and components can be approached in various ways. One example would be a series of cross-layer reliability

interfaces where lower-level components export information about their internal faults or performance deviations and higher-level components use this information to either tolerate such problems or propagate them to higher system levels. Another approach would be a system-wide monitor that would aggregate information across layers and use statistical analysis to coordinate maintenance actions or predict future failures. Such tools are expected to improve system performance and maintainability as well as to significantly reduce the time to achieve full system stability (currently around one year) by identifying marginal components and making it possible to fully utilize the system even before all low-level reliability issues have been addressed.

This infrastructure will have direct implications on the other challenge problems. If the system can export information about the correctness of its results, it is possible to develop novel algorithms that take advantage of such data to improve their reliability or performance. For example, an algorithm that was informed that a given block of memory is unreliable may choose to use it as a cache rather than as primary working space.

Challenge 2: *Enable users to reduce the vulnerability of systems and applications to faults*

In implementing reliable systems and applications, developers are faced with the daunting challenge of identifying and handling a wide variety of system failures and their many possible manifestations. Although an improved understanding of the effect of faults on systems is a critical part of this process, it still leaves developers with a very difficult development task. First, many conventional algorithms are brittle in the face of failures and must be replaced by new variants that are resilient to failures. For example, physical simulations are frequently very tightly coupled, making them very sensitive to any load imbalance or timing variation anywhere in the system (*e.g.* variation due to performance degradation). In particular the POP ocean model slows down by 30%–1600% in the presence of such variation. Second, many programming models currently available do not help developers to create reliable applications, either because they provide features that make applications more brittle or because they are missing features that may simplify the development of reliable applications. As an example of the former, the common shared-memory programming model reduces application reliability by encouraging frequent fine-grained accesses to any byte in memory, which makes it expensive to track and detect errors and encourages applications to become tightly-coupled and therefore sensitive to timing variation. An example of the latter is the fact that conventional programming models are missing features such as robust and efficient checkpoint/restart capability and integrated invariant specification and checking.

As reliability becomes an even more critical component of application development, there is a pressing need for novel reliable algorithms and programming models to improve developers' ability to write reliable applications and systems.

When considering the development of reliable applications, an application's fault vulnerability can be expressed in terms of three components:

- Temporal Sensitivity—dependence by one part of the application on the amount of time taken by another part
- Interaction Locality—the amount and granularity of interactions between application threads and components
- Correctness Sensitivity—the sensitivity of the algorithm to errors in the computation

All three types of locality are individually important to the algorithm's ability to tolerate failures. Applications with good temporal sensitivity will be minimally affected by failures that merely result in timing variation and will be easy to combine with techniques like checkpoint/restart that convert

failures into timing variations. In contrast, applications with poor temporal sensitivity will suffer from severe performance degradations due to failures. Interaction locality is critical for limiting the amount and granularity of data exchanged by system components. Such constraints slow the spread of data corruptions through the application and enable designers to use more expensive and accurate fault detection/correction techniques at component interaction sites. Finally, good correctness sensitivity reduces the probability that a given failure will corrupt the final application output and reduces the severity of such corruptions.

As a specific example of how algorithmic research can play a role, a natural-language processing application might rely on an algorithm that, instead of updating a single centralized model as new training data arrives, is able to apply independent updates to separate models and periodically merge or synchronize the models to avoid model drift. Understanding the rate of model drift, or proving bounds on it based on the interval between model synchronizations, are problems that can be addressed by new algorithmic research.

From the programming systems point of view, certain languages and programming models either encourage or discourage various types of locality. For example, since Map-Reduce and Linda decouple data producers and consumers, they encourage the development of applications with good temporal sensitivity. Similarly, message-passing and token-passing dataflow encourage spatial locality by forcing users to explicitly identify all their communication channels. In contrast, since shared memory allows applications to interact at the granularity of individual memory addresses and synchronize using fine-grained primitives, it encourages poor interaction locality and temporal sensitivity. Finally, while numerical analysis theory provides numerical applications with ways to reduce their sensitivity to data corruptions, the same is not available for many other application domains such as databases.

Challenge 3: *Measure the reliability and fault vulnerability aspects of real systems*

While it is clearly important to enable developers to create reliable systems, this will be of limited use unless users can verify that a given system actually has the reliability properties they need. For example, if a user is considering purchasing hardware that has a certain mean time between failures and some set of common failure modes, they need to find a software stack that provides the highest level of productivity (balance of performance, cost and reliability) in that environment. Further, once a system is installed, users need mechanisms to empirically measure the system's response to faults to make it possible to effectively manage the system and tune its reliability and performance. This capability is of special importance in the context of leading edge supercomputing systems that take as much as a year to bring up to "production" status.

The use of reliable systems will require tools to empirically and independently measure system reliability.

To this end we will need to focus our efforts on developing suites of benchmark applications, reliability metrics and physical test environments that can evaluate the reliability of systems and applications with respect to the metrics of Time, Energy and Correctness. Such criteria will make it possible to make general statements about the fitness of individual systems, system components and applications for specific tasks and specific physical environments. It would also allow the industry to track its progress towards providing highly productive operation.

Learn More `http://www.relxlayer.org`

Acknowledgment This material is based upon work supported by the National Science Foundation (NSF) under Grant 0637190 to the Computing Research Association (CRA) for the Computing Community Consortium (CCC). The opinions, findings, conclusions, and recommendations expressed in this material are those of the study leaders and do not necessarily reflect the views of the NSF, CRA, CCC, or the employers of the study leadership or participants.

Contact E-Mail: `relxlayer@relxlayer.org` Document release number: LA-UR-XX-XXXXX