

# Guilty As Charged: Computational Reliability Threats Posed By Electrostatic Discharge-induced Soft Errors

Keven Feng, Sandeep Vora, Rui Jiang, Elyse Rosenbaum, and Shobha Vasudevan

Department of Electrical and Computer Engineering  
University of Illinois at Urbana-Champaign, 1308 W. Main St. Urbana, IL 61801 USA  
Email: {klfeng2,sgvora2,rjiang3,elyse,shobhav}@illinois.edu

**Abstract**—Electrostatic discharge (ESD) has been shown to cause severe reliability hazards at the physical level, resulting in permanent and transient errors. We present the first analysis of the effects of ESD-induced errors on instruction-level computation. Our data were measured on a microcontroller test chip fabricated for this study, with discharges from a controlled ESD gun. cosmic-ray-induced soft errors have been widely researched, and modeled as single event upsets (SEUs). Our observations across multiple trials on 3 test chips show that in contrast to radiation-induced errors, ESD can cause much more widespread errors than SEUs. In our trials, we observed system hangs and clock glitches which are serious errors. We also observed errors in the following categories: multiple-bit corruptions across multiple registers, multiple-bit corruptions in the same register, and single-bit corruptions across multiple registers. At the instruction level, these errors manifest as system hangs or serious malfunctioning of I/O operations, interrupt operations, and data/program memory. We demonstrate that ESD-induced errors form a significant reliability threat to higher-level functionality, warranting modeling and mitigation techniques.

## I. INTRODUCTION

Electrostatic discharge, or ESD, is the sudden flow of current resulting from the approach of an electrically charged object toward a grounded conductor. The sudden burst of current may generate noise via magnetic coupling or, if the power dissipation is high, damage a micro-scale component of the system. ESD is known to be a major reliability concern at the physical level, resulting in permanent failures as well as transient errors [1]–[3]. Nevertheless, the effect of ESD-induced soft errors has not been studied beyond the physical level. In this work, we present strong evidence that ESD-induced noise can propagate to the higher levels of a stack and cause widespread corruption of registers, affecting the program state. Our measurements were made on a dedicated test chip, instrumented with detection capabilities, under a controlled laboratory environment for injecting ESD current into each chip. We analyze the effects of the generated ESD events at the register level and the instruction level, using several error detection and diagnosis procedures.

### A. Motivation

ESD current can enter a system at breaks in the product casing, such as connectors, push buttons, and displays, and may then propagate to the exposed pins of an integrated circuit (IC). ICs with external pins are most vulnerable to system-level ESD; an external pin has a direct connection to the outside world, e.g., a USB connector. An external pin may be protected by a transient voltage suppressor at the system-level, and only a residual current will enter the pin [4]. In

that situation, the IC will be protected against hard failure, but that will not prevent the residual current from generating noise and soft errors. Alternatively, the external pin may be designed to self-protect against the large, system-level ESD current. With ones to tens of amperes entering the IC, a huge amount of noise is generated. Even an IC without any external pins may experience ESD-induced soft errors. The time-derivative of the system-level ESD current has a value in excess of  $10^{10} A/s$ , resulting in generation of noise via magnetic coupling to signal traces in the system. The corrupted signals enter the IC, possibly resulting in errors.

While particle-induced soft errors have received a lot of attention, and their effects have been studied at all levels of the stack [5]–[8], ESD-induced soft errors have hitherto not been investigated for effects at the higher levels of computation. An important reason for studying these effects at this time is the recent surge of autonomous vehicle technologies. Modern automotive vehicles extensively use microcontrollers and other ICs for subsystems like the engine, transmission, airbags, navigation, braking, collision avoidance, and lane correction. Automotive electronics are exposed to electrostatic discharge and other harsh electrical transients. A hardware transient error that propagates and causes computational errors, especially in autonomous subsystems, can result in serious risks to safety, potentially leading to loss of life and property.

To study the effect of system-level ESD on computations, we designed and fabricated a microcontroller test chip based on the OpenMSP430 in a 130-nm CMOS technology. The microcontroller’s ubiquitous presence in automotive environments that are both ESD-rich and safety-critical makes it a highly relevant test case for our analysis. OpenMSP430 is the open-source version of a widely used commercial architecture.

Our experimental findings reveal that the nature of errors caused by ESD is much more widespread than the randomly occurring single-event upsets (SEUs) used to model particle strike-induced transient errors. In contrast, along with SEUs, ESD-induced errors affect (A) multiple bits in multiple registers, (B) multiple bits in a single register, and (C) single bits in multiple registers. In our trials, we observed that SEUs appeared most frequently at moderate-amplitude, negative precharge voltages; the highest incidence of SEUs, 49.00%, occurred at  $-2.0$  kV. In contrast, for high positive precharge voltages (3 kV and 4 kV), up to 100% of the errors were not SEUs, but belonged to categories A, B, and C. We observed up to 339 bit flips occurred across 82 registers at higher voltages. That has deep implications in the modeling of such events, as well as their potential ability to cause widespread errors.

In our trials, we found catastrophic errors that resulted from bit flips, causing system hangs. We also found serious errors that resulted in incorrect triggering of non-maskable interrupt (NMI) requests, input/output (I/O), and data memory and program memory corruptions. We found that special-purpose registers that interfaced with I/O lines and other external-facing control signals (like the interrupt request, watchdog, debug unit, and monitor registers) were more susceptible to ESD-induced errors than were general-purpose registers (GPRs) that were well-protected within the core. Presumably, the reason is that the special-purpose registers shared the same bus lines that were affected by ESD-induced noise. We also found that ESD caused glitches to the clock, and such glitches could result in catastrophic global errors.

We surmise the existence of silent data corruptions (SDCs) and masked errors from discrepancies in the propagation of register corruptions to program-visible state. To the best of our knowledge, this is the first systematic study that demonstrates the symptoms and effects of ESD-induced errors at higher levels of computation by using actual hardware measurements. In the rest of the paper, we present a detailed analysis of all the symptoms detected and diagnosed because of this important class of errors.

## II. RELATED WORK

A body of work on system-level ESD design has tackled different causes of ESD-induced hard failures [1]–[3]. The body of work on ESD-induced soft errors, i.e., transient errors, is smaller and has focused mostly on physical-level reliability [9]–[11]. In marked contrast the effects of soft errors caused by energetic particle strikes [5]–[8] have been studied extensively at all levels of the program stack. Other work has investigated different techniques for detecting ESD-induced soft errors [12], [13], but hasn't analyzed the impact of these errors and has been limited to cyclic redundancy check (CRC) errors.

In [14], hardware injections made using an ESD gun were made on various commercial processor and microcontroller boards. It was observed that the errors caused by ESD could corrupt applications. These corruptions comprised system hangs, SRAM data corruption, and display errors, among others. That work established causality between ESD hardware injections and application level errors. However, it offered no visibility into commercial boards, and hence no explanation or analysis of how these errors manifest and propagate.

## III. BACKGROUND

### A. ESD-induced errors

The known physical causes of ESD-induced soft errors [10], [15] are described below.

**Glitches:** Glitches are the result of magnetic coupling in a system [10]. The effects of glitches are varied. Glitches on a clock input can cause widespread malfunctioning of the circuit; glitches on a reset pin may cause an inadvertent reset; and glitches on a signal trace may cause logic errors or be ignored, depending on the timing of the noise pulse with respect to the clock edge.

**Supply noise:** When ESD current enters an IC, the path the current takes to return to the board must go through parasitic inductances associated with the power and ground nets of the IC package. This results in a bipolar  $L * di/dt$  noise generated

by the rising and falling edges of the current pulse. The induced noise has both common-mode and differential-mode components. The common-mode component causes glitches at input pins [10]; the incoming signal is referenced to the board ground, while the on-chip ground net bounces with respect to the board ground. The differential-mode component affects the noise margin of circuits on-chip. As supply noise is not localized, many gates and registers may be affected at once.

### B. ESD qualification testing

ESD testing is performed in accordance with the IEC 61000-4-2 ESD test standard [16]. This ESD test is intended to emulate the discharge of a charged operator into a system (end product). The standard prescribes the waveform that should result from contact discharge into a  $2\Omega$  target, as shown in Figure 3, based on the voltage to which the testing equipment is charged. It specifies that the amplitude of the first current peak should be 3.75 A/kV with a rise time of 0.8 ns, and that there should be a current of 2 A/kV at 30 ns and 1 A/kV at 60 ns. This ESD test is used worldwide. Other test procedures, including the ISO:10605 and the DO-160, use waveforms similar to these in the IEC standard for ESD testing. None of these tests require more than 10 discharges to pass [4].

## IV. METHODOLOGY FOR ESD ERROR MEASUREMENTS IN HARDWARE

We fabricated a test chip for our study. The test chip IO ring includes primary and secondary ESD protection, which is typical of industry designs. Details are provided below.

### A. OpenMSP430 test chip

The OpenMSP430 is modeled after the commercial 16-bit MSP430. The chip was designed for an operating speed of 100 MHz; however, synthesis, and place and route, were performed at 125 MHz to allow for margin.

The microcontroller core runs on the 100 MHz clock as well as a low-frequency 32 kHz clock. It has a 27-instruction ISA. It contains a 3-state instruction pipeline. On-chip memory is provided by SRAM, of which 4 kB is program memory (PMEM) and 1 kB is data memory (DMEM).

We inserted a scan chain in the chip during synthesis. The scan chain allows for the read-out and scan-in of data to most of the core and peripheral registers. The internal state of the chip is transparent through the scan chain and gives more detail on the chip-level effects of ESD. It can also facilitate readout of data in the event of an application crash. To use the scan chain, the chip is switched into scan mode, which halts the core and prevents running applications.

The chip layout is shown in Figure 1. Off-chip voltage regulators provide power to the IO circuits (3.3 V) and the core logic (1.2 – 1.5 V). The chip is assembled in a 40-pin quad flat no-leads (QFN) package. All pins are connected to on-chip ESD protection circuits that have been designed to protect the chip against hard failure caused by component and system-level ESD; the ESD design details are given in [17]. Twelve of the IC pins are dedicated to power distribution. Each signal input contains a Schmitt trigger for enhanced noise rejection. In addition, the system reset line contains an on-chip 10 ns low-pass filter to further reduce the likelihood that noise, e.g., an input glitch, will trigger a system reset.

In a custom modification of the OpenMSP430, the test chip includes supply noise monitor circuits [17], which are labeled in Figure 1 as MB0 and MB1.

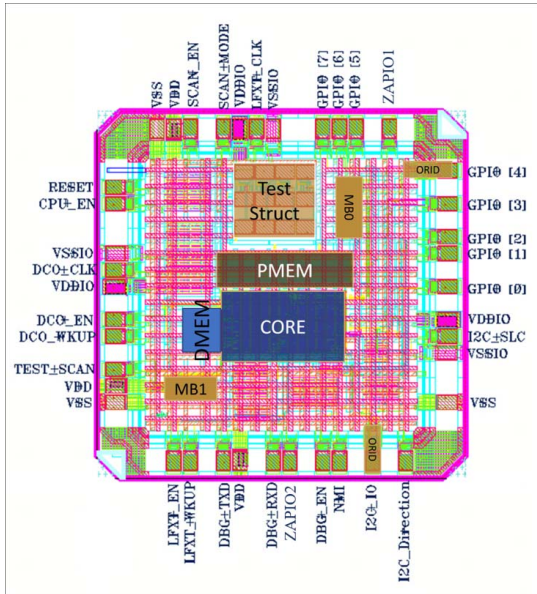


Fig. 1: Layout of the OpenMSP430 test chip.

### B. I/O ring

Electrical pads around the edge of the chip shown in Figure 1 serve as a connection between the pins of the IC and the die. The pins are electrically connected by ultrasonic welding of copper wires (bond wires) onto the top-level metal used. ESD protective elements are placed under the bond pads and prevent the discharge from entering the core. Details of the ESD protection are similar to those described in [10].

## V. INJECTING AND DETECTING ESD ERRORS

We describe 1) the process of generating ESD events by using a physical ESD gun, and 2) our error detection methodology.

### A. ESD event generation

The test setup is shown in Figure 2. The ESD source is called an *ESD gun*. The equipment under test (EUT) is placed on an insulator, which, in turn, is placed on top of a coupling plane (HCP). In this work, contact discharge testing is performed. We refer to an ESD discharge event generated by the gun as a *zap*. Discharges are directed to signal traces (labeled as "ZAPIO1" and "ZAPIO2" in Figure 1).

To facilitate repetitive testing while keeping conditions invariant, we mounted the ESD gun on an automated tester [18]. We performed tests at gun precharge voltages from  $-5$  kV to  $+5$  kV. Those discharge amplitudes were sufficiently low (for our design) that hard failure did not occur. We tested a minimum of three chips, with at least one hundred trials per precharge voltage. The discharge is actuated by a solenoid switch; hence, there was no way to correlate the discharge event with a specific clock cycle during application operation.

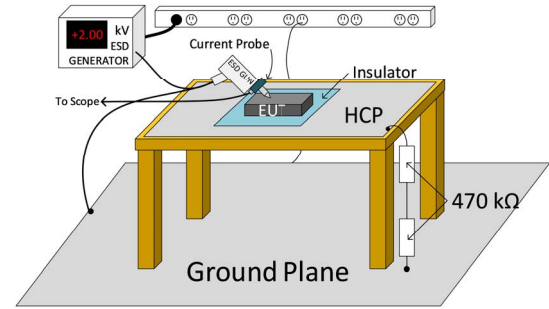


Fig. 2: IEC testbed. The charged elements inside the ESD gun are discharged into the equipment under test (EUT) via the gun tip. Details can be found in [16].

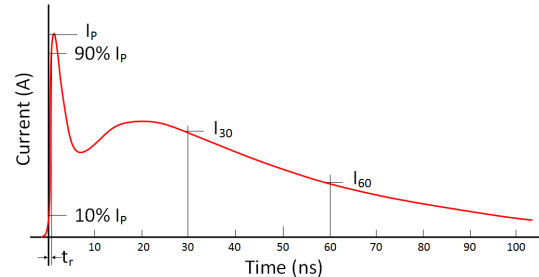


Fig. 3: Current waveform generated by the discharge of the ESD gun into a  $2\ \Omega$  target [16]. The standard stipulates the values of peak current and rise time. The IEC 61000-4-2 test standard also specifies the values of  $I_{30}$  and  $I_{60}$ .

### B. Scan chain testing

We can observe the corruptions to internal registers of the microcontroller that are not normally visible to the user by reading out the values of the internal registers through the scan chain before and after an ESD event. The scan chain connects all the internal registers together into a shift register when in scan mode, which can then be used to shift data out. That shift generates a trace of the chip that can be used to recreate the bit flips as a result of ESD for simulation purposes.

We performed tests on the scan chain to observe corruptions in terms of bit flips in the internal registers resulting from an ESD zap. We initialized the scan chain before the ESD zap to three different initial states as follows: (i) writing all zeros into the scan chain, (ii) writing all ones into the scan chain, or (iii) writing a fixed pseudo-random bit sequence into the scan chain. The scan chain readout is destructive in our design and does not preserve the internal state for normal operation after a scan. Consequently, we are unable to observe silent data corruptions during normal program operation, since the internal register values are unknown. In future works, a redesign to resolve this will let us use our current data for a trace.

The above process is illustrated on the right-hand side of Figure 4. Trials were conducted for precharge voltages from  $-2$  kV to  $-4$  kV and from  $2$  kV to  $4$  kV with intervals of  $1$  kV.

### C. Assembly-level program testing

We write assembly-level programs to act as instruction-level detectors of ESD error symptoms. In our automated error detection setup, a Raspberry Pi controls the ESD gun



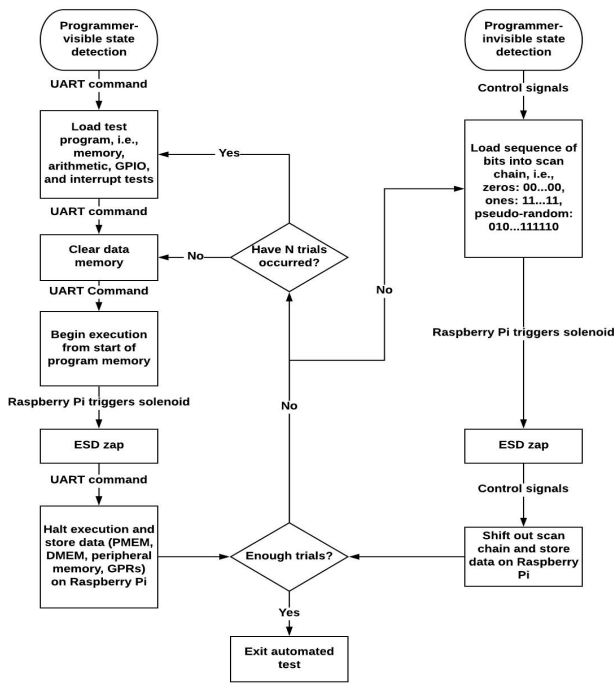


Fig. 4: Block diagram of the automated detection procedures.

and the microcontroller. The Raspberry Pi triggers the ESD gun by running current through a solenoid, a mechanical process that takes variable time. By the time the zap hits the microcontroller, many cycles beyond a desired point may have passed. That makes the timing of the zap difficult to control. Thus, our programs for error detection were structured in such a way that the exact timing of the zap does not affect the error detection. Each program has an initialization phase and an infinite loop that, once corrupted, retains the corrupted value. For example, one way to preserve a value would be to read a value in memory and then write the value read back to memory. Since the data are not overwritten in the lifetime of the program, corruptions are retained, irrespective of the time elapsed before the data are read.

In a single trial, we loaded the detector assembly program onto the microcontroller and executed it for one second. We then applied a controlled zap by the gun. We read the contents of data memory, program memory, peripheral registers, and GPRs onto the Raspberry Pi by using the debug unit of the microcontroller. That process is illustrated on the left-hand side of Figure 4. The peripherals and GPRs data were also part of the scan chain.

We classify our error detectors based on the program visible state components that they check.

- **Memory detectors:** The first detector initializes all of the memory to a predetermined value and does no further memory operations during the ESD event. That helps detect errors that occur in memory that is not in use. Another detector initializes memory and reads during the ESD event in order to detect whether active memory is more susceptible to corruption than memory that is not in use. A third detector does memory write operations during the ESD event.
- **Arithmetic unit and general-purpose I/O (GPIO) de-**

**tectors:** These programs detect errors in the ALUs and GPIOs. The majority of these instructions are the basic arithmetic operations such as ADD, AND, OR, SUB, and XOR. We tested the input and output functionality of the GPIOs separately. Notably, during both types of detection, the debug unit was always affected by the ESD zap, causing a system hang. We recovered results despite that error by storing the results in memory, which did not lose data after a system reset.

- **Interrupt detectors:** These programs check various aspects of interrupts. For all programs, all 16 interrupt vectors performed a unique action in memory without explicit use. In the first test, interrupts were disabled to check if an IRQ can trigger while being disabled. The second test then enabled the interrupts. The final detector checked whether the watchdog interrupt could be triggered by errors because of ESD in the watchdog.

## VI. EXPERIMENTAL RESULTS

### A. Register errors in intermediate states

In Table I, we characterize results in the scan chain by the number of bit flips. Multiple bit flips indicate more serious errors than an SEU. We further categorize multiple bit flips in Table II based on their distribution among different register arrays. A is multiple-bit corruptions over multiple registers; B is multiple-bit corruptions in the same register array; and C is single-bit corruptions across multiple register arrays.

We began to observe errors at only the magnitudes of precharge voltages (outside  $\pm 2.0$  kV). Interestingly, the single-bit errors (Category C) tend to peak around  $-3$  kV and  $2$  kV, but fall sharply at higher precharge voltages. The multiple-bit errors in the same register array (B) occurred starting at  $40.63\%$  at  $-3$  kV and then decreased to  $15.01\%$  at  $-4$  kV. For negative precharge voltages, A was first observed at  $-3$  kV and then increased to  $24.94\%$  at  $-4$  kV. The multiple bit errors (A, B) peaked up to  $100\%$  at precharge voltages higher than  $3$  kV. The number of corrupted bits, as well as the number of registers corrupted, was very high above that voltage, indicating widespread errors.

### B. Errors in program-visible state

**System hangs:** In our trials, the most serious programmatic errors were system hangs. In the first error we observed, the debug unit used to control the microcontroller hung after every ESD zap with a precharge voltage above  $3$  kV or below  $-3$  kV. The cause was errors in the debug state machine, which we could recover from only by a reset. Other errors included program restarts. Program restarts often accompanied memory errors such that program memory corruptions early in the program memory were re-executed, overwriting a task that might have previously been computed correctly. This type of error can be explained by the triggering of the reset IRQ.

Table III shows the percentage of errors that occurred during the execution of an assembly program. The rows indicate the specific components that were being tested during the ESD zap, and the columns show the precharge voltages of the zap.

**Memory errors:** We designed the chip to operate with the supply voltage for its core logic set to any value in the range of  $1.2$  V to  $1.5$  V. Memory errors were observed at a  $1.2$  V supply voltage, but not at  $1.5$  V, over the range of

ESD stress levels (precharge voltages) we investigated. Our observations in Table III are split into data memory errors and program memory errors. When the program is corrupted, it cannot be fixed by a reset and must be reprogrammed by an external controller for restoration. Programmer corruption can thus be considered a permanent failure for a microcontroller, if that solution is not readily available. We observed across many trials that for a given chip, the same memory cells were corrupted. That suggests that a small fraction of memory cells have a low noise margin, perhaps due to process variation. We observed bit flips in the SRAM only when the stress level was raised all the way to +5 kV, indicating that “low” noise margin is a relative term. Presumably, the same bit flips would occur at higher stress voltages, but we could not test that, since at +6 kV, the test chip might suffer from hard failures.

**Register errors:** In our trials, GPRs and ALUs were affected 0% of the time at all precharge voltages. Errors were observed in registers that interfaced with interrupts, debug, and other I/O lines. We found that the interrupt and I/O lines had error rates of up to 98% for a given interrupt and up to 25% for the GPIOs because we directly injected current into the I/O circuitry; in contrast, noise was attenuated before it reached the core domain. The ESD noise propagated to the core through the on-chip ESD protection network and via coupling between on-board traces, which can account for the discrepancy between the GPR error rates and other “uncore” registers error rates in the scan chain results. We observed maximal errors in the external-facing registers, such as the debug UART registers, and the IRQ registers. They tended to be connected to the same bus, and presumably interfaced with the same noisy signals affected by ESD. Registers more visible to I/O tend to be more susceptible to errors. In particular, all of those registers reverted to their reset state at the higher ESD precharge voltages of 3 kV and 4 kV.

Consequently, we observed the erroneous triggering of interrupts due to ESD. We determined that IRQ2, a user-defined interrupt line, was in error over 98% of the time. IRQ14, which is an NMI, was also triggered in error about 20% of the time. Interrupts must be disabled in order to prevent these errors. If the interrupt trigger is coupled with a program memory error that modifies the interrupt vector table, incorrect code may be executed, although we did not observe that error.

### C. Error propagation

In general, we observed a trend in which errors started to appear in programs at the precharge voltages at which there were a large number of average bit flips per trial in the scan chain. We observed, however, a discrepancy between the widespread register errors in the scan chain and the relatively few errors that propagated to the program-visible state. For example, at 4 kV, errors were observed in GPRs, watchdog registers, and the memory address register, among others. Masking could be a reason for the lack of propagation of the observed errors. For instance, watchdog registers were password protected, so a flip might not have sustained in the absence of the correct password. Another distinct possibility is the existence of silent data corruptions (SDCs). Errors might have existed in the chip that did not propagate to programmer visible state for the given programs, and they could present potential corruptions to the system.

Another reason is the lack of controllability of the timing of the zap: the ESD zap could occur during any cycle or stage of the pipeline. For instance, a memory address register flip will have an impact only during a memory access. Before or after the memory access, a flip will not result in an error. It is also possible that our current program detectors have exposed only some of the errors, and that future detectors will reveal more symptoms in the program state.

### D. Clock glitches

Clock glitches are a cause for concern, because a glitch on the clock can result in system-wide issues. We tested for clock glitches by programming a specific state into the scan chain and reading out the stored state after a discharge event. We ensured with that experimental design that any observed bit flips in registers were not the result of corrupted data from inputs, as normal inputs were disabled when the scan chain was operational. Glitches on the clock may result from board-level coupling or noise on the IO power supply that moves the switching threshold of the clock input circuitry. To determine the number of clock glitches seen, we programmed a pseudo-random bit sequence into the scan chain. By shifting the post-ESD scan chain readout to match as closely as possible the input version, we determined the number of clock glitches.

Figure 5 shows that clock glitches occurred as a result of ESD. A single ESD event likely causes multiple clock glitches.

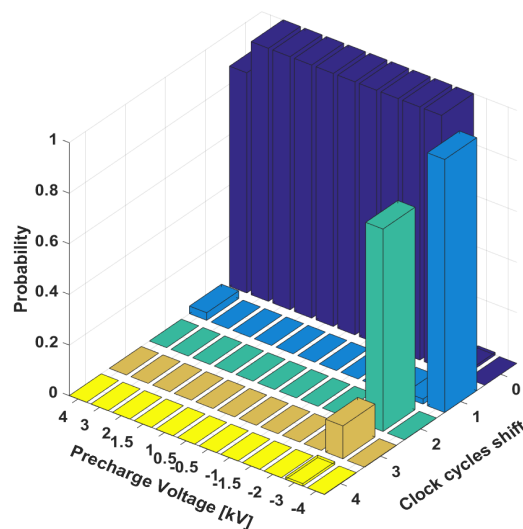


Fig. 5: Clock glitches from ESD based on precharge voltage.

## VII. DISCUSSION AND CONCLUSIONS

Existing high-level detectors cannot inject ESD faults into simulators. SPICE models that have been developed for the ESD gun and I/O ring can be used to simulate these fault environments, while mixed-mode simulation can be used for simulating clock glitches and supply noise to model the ESD effects. In summary, we have presented evidence on real chip measurements that categorically demonstrates the reliability threat posed by ESD-induced errors’ propagation to higher

TABLE I: Aggregated errors in the scan chain of a representative test chip. Discharges performed on ZAPIO1.

Precharge voltage	-4 kV	-3 kV	-2 kV	-1.5 kV	-1 kV	-0.5 kV	2 kV	3 kV	4 kV
Percent with 2+ bit errors	77.00%	63.50%	25.00%	33.33%	33.67%	33.67%	0.00%	85.00%	100.00%
Percent with 1 bit error (SEUs)	23.00%	36.25%	49.00%	0.00%	0.67%	29.00%	0.33%	15.00%	0.00%
Percent with 0-bit errors	0.00%	0.25%	26.00%	66.67%	65.67%	37.33%	99.67%	0.00%	0.00%
Average bit errors	1.70	1.89	1.05	0.67	0.68	0.98	0.00	280.40	339.22
Average registers with bit errors	1.03	1.16	1.05	0.67	0.68	0.98	0.00	70.59	82.02

TABLE II: Distribution of multiple flips from Table I. 2kV is N/A as there were no multiple-bit error trials.

Precharge voltage	-4 kV	-3 kV	-2 kV	-1.5 kV	-1 kV	-0.5 kV	2 kV	3 kV	4 kV
A: Multiple errors across registers	24.94%	0.65%	0.00%	0.00%	0.00%	0.00%	N/A	59.48%	59.56%
B: Multiple errors on one register	15.01%	40.43%	0.00%	0.00%	0.00%	0.00%	N/A	0.00%	0.00%
C: Single error across registers	60.05%	58.92%	100.00%	100.00%	100.00%	100.00%	N/A	40.52%	40.44%

TABLE III: Program errors of a representative test chip. Discharges performed on ZAPIO2.

Precharge voltage	5 kV	4 kV	3 kV	-3 kV	-4 kV	-5 kV
I/O errors	25.00%	25.00%	25.00%	25.00%	25.00%	25.00%
Interrupt errors	99.00%	99.33%	99.00%	100.00%	100.00%	100.00%
GPR errors	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Data memory errors	9.67%	0.00%	0.00%	0.00%	0.00%	0.00%
Program memory errors	21.00%	0.00%	0.00%	0.00%	0.00%	0.00%

levels of the stack. We argue that there should be an effort to model this important category of errors and simulate their effects for larger, more complex systems than the one we have investigated. Techniques for mitigation of and/or recovery from these errors should also be developed.

#### ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 1526106. The authors would like to thank Mr. O. Girard for design of the open source core and useful discussions. We also thank Dr. Karan Bhatia of Texas Instruments for discussions about microcontroller design.

#### REFERENCES

- [1] E. A. Amerasekera and C. Duvvury, *ESD in Silicon Integrated Circuits*. John Wiley & Son, 2002.
- [2] C. Duvvury and H. Gossner, *System Level ESD Co-design*. John Wiley & Sons, 2015.
- [3] Y. Cao *et al.*, "ESD simulation with Wunsch-Bell based behavior modeling methodology," in *Proc. Electrical Overstress/Electrostatic Discharge Symposium (EOS/ESD)*. IEEE, 2011, pp. 1–10.
- [4] JEDEC, *System Level ESD, Part 1: Common Misconceptions and Recommended Basic Approaches*. JEDEC, 2011.
- [5] E. Cheng *et al.*, "Clear: Cross-layer exploration for architecting resiliency: Combining hardware and software techniques to tolerate soft errors in processor cores," in *Proc. Design Automation Conference (DAC)*. IEEE, 2016, pp. 1–6.
- [6] P. Ramachandran *et al.*, "Hardware fault recovery for I/O intensive applications," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 11, no. 3, p. 33, 2014.
- [7] S. K. S. Hari *et al.*, "GangES: Gang error simulation for hardware resiliency evaluation," in *Proc. International Symposium on Computer Architecture (ISCA)*. IEEE, 2014, pp. 61–72.
- [8] K. Pattabiraman *et al.*, "SymPLFIED: Symbolic program-level fault injection and error detection framework," in *Proc. International Conference on Dependable Systems and Networks with FTCS and DCC*. IEEE, 2008, pp. 472–481.
- [9] N. Thomson *et al.*, "Custom test chip for system-level ESD investigations," in *Proc. Electrical Overstress/Electrostatic Discharge Symposium (EOS/ESD)*. IEEE, 2014, pp. 1–10.
- [10] N. Thomson, Y. Xiu, and E. Rosenbaum, "Soft-failures induced by system-level ESD," *IEEE Transactions on Device and Materials Reliability*, vol. 17, no. 1, pp. 90–98, 2017.
- [11] K. Domanski and H. Gossner, "Soft fails due to LU stress of virtual power domains," in *Proc. Electrical Overstress/Electrostatic Discharge Symposium (EOS/ESD)*. IEEE, 2015, pp. 1–8.
- [12] P. Maheshwari *et al.*, "Software-based analysis of the effects of electrostatic discharge on embedded systems," in *Proc. Computer Software and Applications Conference (COMPSAC)*. IEEE, 2011, pp. 436–441.
- [13] P. Maheshwari *et al.*, "Software-based instrumentation for localization of faults caused by electrostatic discharge," in *Proc. High-Assurance Systems Engineering (HASE) International Symposium*. IEEE, 2011, pp. 333–339.
- [14] S. Vora *et al.*, "Application level investigation of system-level ESD-induced soft failures," in *Proc. Electrical Overstress/Electrostatic Discharge Symposium (EOS/ESD)*. IEEE, 2016, pp. 1–10.
- [15] B. Orr *et al.*, "Analysis of current sharing in large and small-signal IC pin models," in *Proc. Electrical Overstress/Electrostatic Discharge Symposium (EOS/ESD)*. IEEE, 2014, pp. 1–6.
- [16] "Electromagnetic compatibility (EMC) - part 4-2: testing and measurement techniques - electrostatic discharge immunity test (IEC 61000-4-2)," NSAI, 2008.
- [17] S. Vora *et al.*, "Hardware and software combined detection of system-level ESD-induced soft failures," in *Proc. Electrical Overstress/Electrostatic Discharge Symposium (EOS/ESD)*. IEEE, 2018, pp. 1–10.
- [18] Y. Xiu *et al.*, "Stochastic modeling of air electrostatic discharge parameters," in *Proc. International Reliability Physics Symposium (IRPS)*. IEEE, 2018, pp. 2C.2\_1–2C.2\_10.