SAND2017-0526PE

Robust Digital Computation in the Physical World

Jackson R. Mayo
Sandia National Laboratories, Livermore, CA 94551, USA
jmayo@sandia.gov

Sandia National Laboratories is a multi-mission laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Acknowledgments

- Key collaborators at Sandia: Robert C. Armstrong, Geoffrey C. Hulette, Maher Salloum, Andrew M. Smith
- This talk includes material presented at
  - 2014 Workshop on Numerical Software Verification
  - 2015 IEEE Systems Conference
  - 2015 Workshop on Formal Techniques for Safety-Critical Systems
  - 2016 Workshop on Fault Tolerance for HPC at Extreme Scale

Begin at the beginning

- What is a digital system?
- Working definition:
  A physical system designed to have discrete combinatorial states and to perform information processing

Begin at the beginning

- What is a digital system?
- Working definition:
  A physical system designed to have discrete combinatorial states and to perform information processing
    - physical system: ultimately subject to contingencies and uncertainties like other physical systems – it's amazing how often we can ignore this but how difficult things get when we can't

Begin at the beginning

- What is a digital system?
- Working definition:
  A physical system designed to have discrete combinatorial states and to perform information processing
  - physical system: ultimately subject to contingencies and uncertainties like other physical systems – it's amazing how often we can ignore this but how difficult things get when we can't
  - discrete: typically Boolean, the states are "attractors" for the underlying continuous physics (e.g., 0 and 5 volts) and make digital systems deterministic and lossless *in many cases*

Begin at the beginning

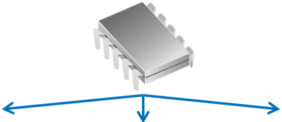
- What is a digital system?
- Working definition:
  A physical system designed to have discrete combinatorial states and to perform information processing
  - physical system: ultimately subject to contingencies and uncertainties like other physical systems – it's amazing how often we can ignore this but how difficult things get when we can't
  - discrete: typically Boolean, the states are "attractors" for the underlying continuous physics (e.g., 0 and 5 volts) and make digital systems deterministic and lossless *in many cases*
  - combinatorial: a large number of elements can change independently, creating vast combinations to store information (N bits give 2^N states)

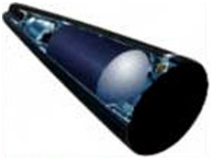
Begin at the beginning

- What is a digital system?
- Working definition:
  A physical system designed to have discrete combinatorial states and to perform information processing
  - physical system: ultimately subject to contingencies and uncertainties like other physical systems – it's amazing how often we can ignore this but how difficult things get when we can't
  - discrete: typically Boolean, the states are "attractors" for the underlying continuous physics (e.g., 0 and 5 volts) and make digital systems deterministic and lossless *in many cases*
  - combinatorial: a large number of elements can change independently, creating vast combinations to store information (N bits give 2^N states)
  - information processing: transforming discrete inputs into discrete outputs using logic operations

Systems engineering raises the stakes

- Sandia missions use digital systems to control and simulate high-consequence physical systems
  - Digital hardware and software are coupled with these other systems, forming high-consequence cyber-physical systems





Weapon controllers

Networked infrastructure

Extreme-scale simulation

Mathematics shows the limits of understanding logic

- Theorem (Turing 1936, Rice 1953): No algorithm exists to predict a priori the behavior of a generic information processing system
  - i.e., such a system is undecidable even if deterministic
  - Practical significance: A real system, with a finite exponentially large number of states but otherwise generic, is *effectively* undecidable – in particular, testing cannot tell us all its possible behaviors
  - We need to bound all possible behaviors to quantify safety and security
- Further complication: Digital systems are also physical
  - We have to deal with "rare events" where logic isn't the whole story

What is the solution space?

- Formal methods (reduced complexity)
    - Automated reasoning about all possible behaviors within a model – widely used in industry
    - Model checking, theorem proving
    - Scaling limitations, though power and tractability have improved over time
- Complex systems theory (structured complexity)
    - Probabilistic analysis of response of networks to perturbations
    - Well suited to understand emergent system-level robustness, but only sparingly applied to engineered digital systems
- In both strategies, systems must be constrained to be analyzable
    - Ideal approach is to consciously design-in analyzability and robustness along with functionality

Careful consideration is needed to verify digital computations interacting with continuous physics

- In many applications, real numbers are not only represented digitally but are also present as actual continuous dynamics coupled via transducers – forming a *hybrid* or *cyber-physical* system
- Most existing formal methods apply to purely digital systems
- Formally modeling and analyzing hybrid systems is an important challenge
  - Need to ensure models are physically consistent and well-posed
  - Need to reason flexibly about continuous and discrete state spaces
- Here we discuss a theorem-proving approach that captures key aspects needed for more powerful reasoning about hybrid systems

Buridan's Principle constrains analog-digital interaction

- All known physical processes have continuous dependence on initial conditions
    - The same should hold for any physical implementation of digital logic
- Thus a *continuous* input at time t_j cannot be guaranteed to result in a discrete decision at *any* finite later time t_i
    - By the intermediate value theorem, there is *some* (perhaps unlikely) range of states at t_j that leaves the system still undecided at t_i – e.g., partway between digital 0 and 1
    - This is Buridan's Principle (Lamport 1984)
    - The presence of random noise does not change the argument – there is still a finite probability to remain in an intermediate state

Buridan's Principle can be interpreted from different viewpoints on physics

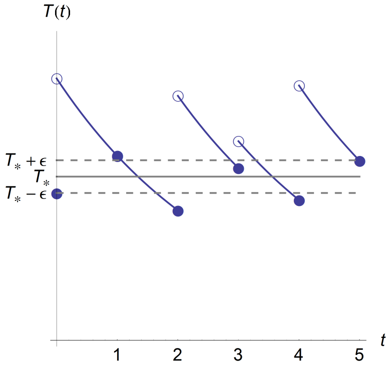
- Usual viewpoint: digital as an abstraction of analog reality
  - As far as is known, all computers are physically continuous
  - Buridan's Principle can be neglected for "purely digital" devices with disjoint, clock-synchronized states (as assumed in most formal methods), but becomes important upon interaction with external continuous or asynchronous processes
- Alternate viewpoint: analog as an abstraction of digital reality
  - It has been speculated that the physical universe may consist of discrete computation (Zenil ed. 2012)
  - Real numbers then idealize a limit of increasingly fine-grained observations, analogous to arbitrary-precision arithmetic
  - Buridan's Principle then corresponds to the Table Maker's Dilemma – unbounded computational time for comparison of a real number to a threshold (or more generally, evaluation of any discontinuous function)

An idealized hybrid system illustrates modeling issues

- Consider a thermostat designed to maintain an object's temperature T in a desired range above ambient temperature
  - Gain from "instantaneous" heat pulse: applied at uniform time intervals if T is below a threshold
  - Loss to environment: linear cooling law
- Buridan's Principle says no device can *guarantee* that either a *full* heat pulse or *none* is applied at a specific time
  - This example can tolerate indecision because, when either a full heat pulse or none is acceptable, an intermediate amount is also acceptable

An idealized hybrid system illustrates modeling issues

- Mathematical description consists of temperature $T : \mathbb{R}_{\geq 0} \to \mathbb{R}$, “arbiter” $\tilde{\theta} : \mathbb{R} \to \mathbb{R}$, and parameters $\alpha, H, T_*, \epsilon \in \mathbb{R}_{>0}$
- Arbiter approximates unit step function: bounded between 0 and 1, with $\tilde{\theta}(\Delta) = 1$ for $\Delta > \epsilon$, and $\tilde{\theta}(\Delta) = 0$ for $\Delta < -\epsilon$



- For $n \in \mathbb{N}$, given $T(n)$ as the temperature just *before* a potential heat pulse at time n, the temperature evolves causally as

$$T(t) = \Big(T(n) + H\,\tilde{\theta}\big(T_* - T(n)\big) \Big)\, e^{-\alpha(t-n)} \quad \text{for all } t \in (n, n+1]$$

Bounds on temperature can be proved informally

- Seek a guarantee on thermostat performance: maintaining the temperature in a range $[A, B]$ with $0 < A < B < \infty$

$$\text{If } \ T(0) \in [A, B], \ \text{ then } \ T(t) \in [A, B] \ \text{ for all } t \in \mathbb{R}_{\geq 0}$$

- This will follow by induction if the following holds for all $n \in \mathbb{N}$

$$\text{If } \ T(n) \in [A, B], \ \text{ then } \ T(t) \in [A, B] \ \text{ for all } t \in (n, n+1]$$

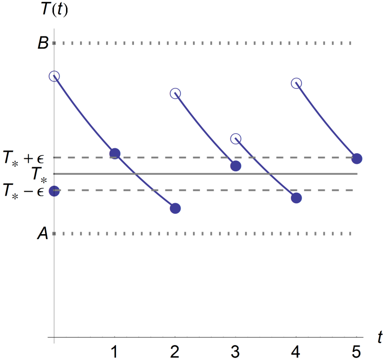
- Given the constraints on the arbiter $\tilde{\theta}$, we can show the property holds provided

$$0 < A \leq \min\left(\frac{H}{e^{\alpha} - 1}, (T_* - \epsilon)e^{-\alpha}\right) \ \text{ and } \ B \geq T_* + \epsilon + H$$

Bounds on temperature can be proved informally

- Given $T(n) \in [A, B]$, inequality algebra demonstrates the bounds by case analysis, ensuring that $[A, B]$ maps into itself, despite the indecision margin ϵ where the heat pulse is indeterminate

- Prove that $T(t) \geq A$ for all $t \in (n, n+1]$: Heat pulse is sufficient if $T(n) < T_* - \epsilon$, and is unnecessary if $T(n) \geq T_* - \epsilon$
- Prove that $T(t) \leq B$ for all $t \in (n, n+1]$: Heat pulse is harmless if $T(n) \leq T_* + \epsilon$, and is absent if $T(n) > T_* + \epsilon$



Higher-order logic can consistently combine discrete and continuous elements

- In higher-order logic, discrete systems are usually modeled inductively, i.e., via well-founded recursive definitions
- Continuous systems are modeled with real numbers, which cannot be defined inductively
  - This often leads to confusion in digital approximations of continuous systems
  - Inductively defined approximations to reals (e.g., floating point) introduce a logical mismatch between the “real” model and the discrete one
  - Proofs on approximations of continuous systems are usually difficult or impossible, and may also prove the wrong things
- Instead, we define real numbers axiomatically
  - Of course we can't “construct” all real numbers in this system
  - But we can reason about them without the problems of approximation

Buridan's Principle is reflected in the formal analysis

Coq definition of $\tilde{\theta}$

```
Parameter eps : R.
Parameter theta_tilde : R → R.
Hypothesis theta_tilde_bound : ∀ d, 0 ≤ theta_tilde d ≤ 1.
Hypothesis theta_tilde_1 : ∀ d, d > eps → theta_tilde d = 1.
Hypothesis theta_tilde_0 : ∀ d, d < -eps → theta_tilde d = 0.
```

- Coq lets us mix definitions using (axiomatic) real numbers with our (inductive) formulation of the discrete system
- Notice that hypotheses *theta_tilde_0* and *theta_tilde_1* involve decisions on comparisons of real numbers
- Even though the comparison is computationally undecidable, it is nonetheless easily provable via axioms

Bounds on temperature can be proved in Coq

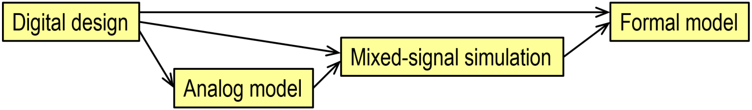
- Given our definitions – temperature computation and continuous physical environment, we can show that our system will keep the temperature within some (continuous) bounds

Formal proof: Temperature is bounded

```
Theorem T_in_interval (Tn tau : R) (tau_bnd : 0 ¡ tau ≤ 1) :
  A ≤ Tn ≤ B → A ≤ T Tn tau tau_bnd ≤ B.
Proof.
  intros HAB. decompose record HAB. split.
  destruct (Rlt_le_dec Tn (Tstar - eps)).
    apply Tn_heat_keeps_above; auto.
    apply Tn_no_heat_keeps_above; auto.
  destruct (Rle_lt_dec Tn (Tstar + eps)).
    apply Tn_heat_keeps_below; auto.
    apply Tn_no_heat_keeps_below; auto.
Qed.
```

Systems analysis can incorporate out-of-nominal electrical behavior

- Research is extending digital systems analysis to address physical environments where a device is not fully digital anymore
- Mixed-signal simulation can elucidate the digital imprint (e.g., bit flip pattern) of a physical insult (e.g., radiation) on a circuit
    - Using analog electrical model for the part of the circuit subjected to the insult
- By including digital upsets in a formal or complexity model, effect on rest of the digital state space can be quantified and mitigated
    - Example: Does a digital safety property still hold even in an accident scenario?



Broader principles support robustness in complex systems

- Biological and social complex systems typically are *not* formally verified, but show impressive robustness to unforeseen failures
- Why? They have inherent stability constraints from their origins in adaptation and selection

- Our hypothesis: Digital designs constrained by formal methods also exhibit enhanced robustness to unforeseen failures by a similar mechanism

Digital system properties directly proved by formal methods are limited

- Guarantees are limited to requirements explicitly encoded by the developer
  - The developer must formally describe the specific "undesired behaviors" in advance
  - A formal tool can then verify the absence of such behaviors over a vast state space (when tractable)
- Guarantees are valid only within the semantics of the system model
  - There may be vulnerabilities in the real system not accounted for in the model (e.g., physical attacks)

Outsize benefits of up-front formal modeling have been noted in practice

- Key observation: Design for analysis yields increased robustness, regardless of *when* or even *whether* the analysis is performed
  - Faults and vulnerabilities are reduced if the developer starts with a high-level formal model – even if no further verification is done and even if the implementation is not explicitly constrained (Woodcock et al. 2009)
  - This supports our hypothesis that robustness is conferred because of design characteristics promoted by the formal modeling process
- By contrast, formal verification *after the fact* does not increase robustness more broadly, if the design was not formally informed

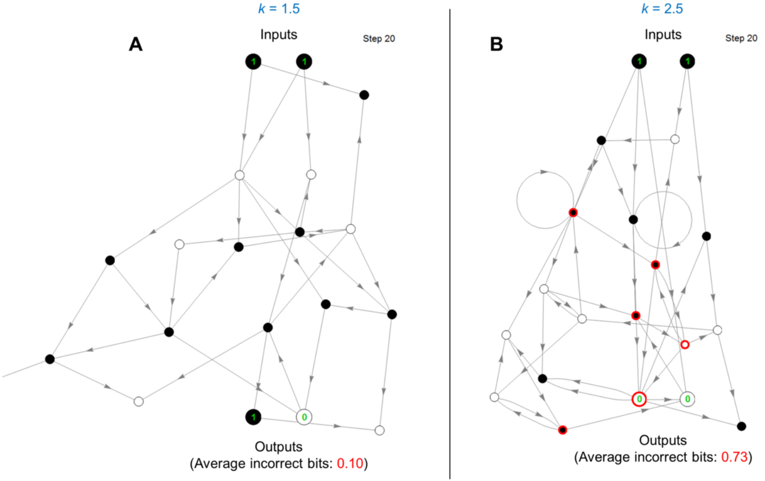
Complex adaptive dynamical systems offer a useful perspective on hardware and software

- As dynamical systems, today's typical digital designs are *chaotic*
- Formal methods, by contrast, enforce *bounded* behavior, similar to that seen in complex systems adapted to their environments
  - To be useful (engineering) or viable (evolution), an adaptive dynamical system must show a coherent response, neither strongly overdamped/inert nor profoundly chaotic/random
  - At the "edge of chaos" (critical) or somewhat below it (subcritical), broad robustness to perturbations is obtained
  - Subcriticality or "smoothness" generalizes the constraints imposed by formal analyzability
- Restricted programming models also extend the power of testing
  - New programming models with intrinsic smoothness could enable more confident generalization of correctness to untested inputs
  - Empirically, incidence of vulnerabilities does differ measurably based on programming language

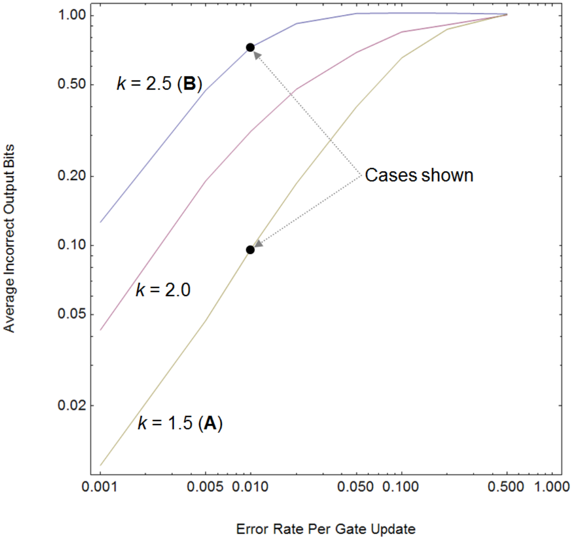
Boolean networks provide a simple representation of digital logic

- Originally investigated in biology, Boolean networks (BNs) correspond closely to hardware sequential logic gates
  - Each node in the directed graph has two possible states, 0 and 1
  - A node's state transition at each discrete time step is determined from its input connections by a "transfer function"
- Create BNs that add two 1-bit numbers (half-adder function), by random sampling and selection
  - This function is very simple, but we seek BNs representative of more complex implementations
  - BN ensembles differ in average inputs per node (k)
  - Select 20-node BNs that compute the correct result for all inputs when operating *nominally*, and then introduce 1% *bit errors* to evaluate robustness
  - Cascading errors are outlined in red

Boolean network "programs" exhibit quiescence for $k < 2$ and chaos for $k > 2$



Ensemble simulations indicate systematic relations of design parameters to robustness



Formal verification confirms insights from dynamical systems theory

- While BN stability is relevant well beyond the reach of exhaustive verification, the example half-adder BNs are simple enough to check directly with formal methods
- With the NuSMV model checker, we exhaustively prove/disprove correct function of these two BNs in the presence of bit errors
  - Using a nondeterministic model that allows any single bit error during a range of time steps
  - Example correctness requirement for carry bit: `LTLSPEC F ((clock=20) & (n18 = (n00&n01)))`
- NuSMV results: Chaotic BN is susceptible to corruption from *any* time step, whereas quiescent BN can be corrupted *only* in the last 5 of 20 time steps and is self-healing otherwise

Failure modes can be understood via abstractions

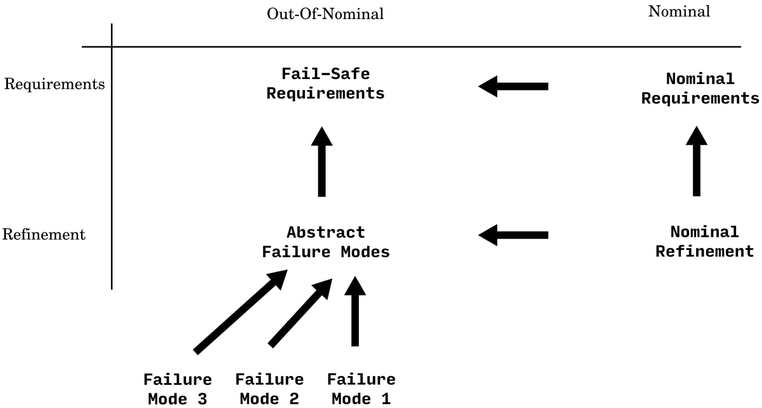
- Examples of failures that result in an overapproximation:
  - A logic gate becomes unreliable and nondeterministic
  - A sensor fails, providing random input to a digital control
  - Generally: any malfunction that generates additional behaviors that were not part of the design intent

- Errors induced by environmental physics are common:
  - Radiation (cosmic rays, etc.)
  - Heating (fire, etc.)
  - Physical insult (destruction of sensor, etc.)
- Abstraction techniques can reveal failure modes for which a particular design will be robust
- Abstraction techniques can support designed-for failure modes anticipating likely accidents and faults

Out-of-nominal requirements abstract nominal requirements

- Assumption: The critical safety requirements imposed in out-of-nominal environments are weaker than (e.g., a subset of) the nominal requirements
- Rationale: Risk management focuses on preventing *catastrophic* failures across the board, but can accept some lesser failures in (what are expected to be) *uncommon* out-of-nominal environments
- Not considered in detail here:
  - Fail-safe design requirements that apply *only* to out-of-nominal behavior
  - Specific methodology for fail-safe design, though plenty of extant methodologies seem applicable

Square diagram shows refinement relationships that preserve requirements



Out-Of-Nominal

Nominal

Requirements

Refinement

Fail-Safe Requirements

Nominal Requirements

Abstract Failure Modes

Nominal Refinement

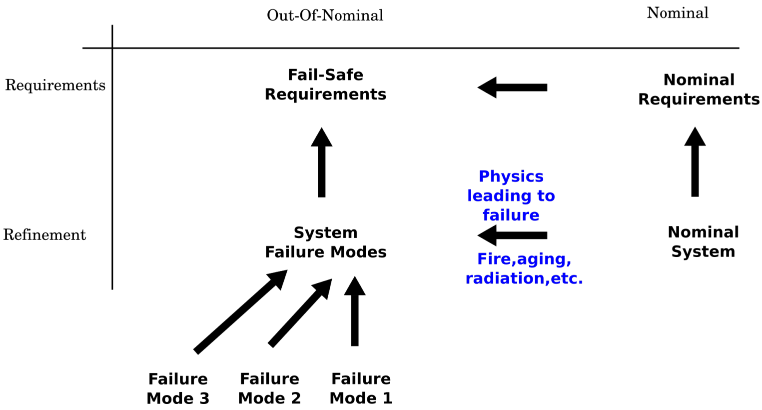
Failure Mode 3

Failure Mode 2

Failure Mode 1

- Refinement/abstraction conceptual diagram for treating out-of-nominal and nominal models in a unified way
- Arrows point in the direction of abstraction

Each abstraction *may* also represent a designed-in response to an abnormal environment



- The physical processes that lead to failure must be modeled to validate their consistency with the abstract failure modes

Existing abstractions reveal in what ways a system is robust

- If abstractions used in proving safety properties for the nominal design (e.g., via CEGAR) can be reinterpreted as a manifestation of faults, then this:
    - Gives the digital designer an idea of what out-of-nominal conditions the system is robust to – for free
    - Suggests that the design can be intentionally engineered to preserve critical safety properties for anticipated failure modes

A supercomputer is itself a complex system with out-of-nominal behavior

- High-performance computing (HPC) faces a resilience problem
  - Sheer scale (hundreds of thousands of processors) magnifies previously negligible hardware errors even for a correct program in a nominal environment
- Physics simulation (main HPC application) is a highly non-generic program; we can take advantage of its structure and smoothness
  - Numerical analysis already addresses stability to truncation errors
  - Idea: Extend the mapping between the digital computation and the physics being simulated, so that the computation gains similar inherent stability to faults
  - An instance of algorithm-based fault tolerance
- Analogy between extreme-scale HPC and small-scale remote/portable embedded computing: Both are typically power-constrained

Problem: Future HPC platforms will face tradeoffs imperiling correct hardware function

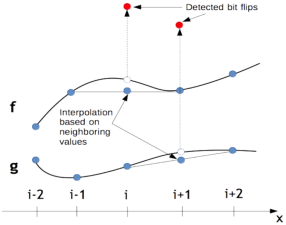
- Hardware correction already attempts to hide many "out-of-nominal" behaviors from the application
    - Error correction for bit flips in memory and caches is important and largely effective
- Increasing scale and constrained power may push toward exposing *silent* hardware errors (of possibly unexpected kinds) – corrupting an unaware application's results
- A primary concern is silent data corruption (SDC), where the computation appears normal except for wrong numerical values
    - Undetected memory errors at exascale (10^{18} Flops) for one type of error-correcting (ECC) memory could be $\sim$1 per day
    - Low-voltage processors and accelerators will likely have increased rates of arithmetic errors; ECC doesn't protect data transformation

Objective: Enable practical SDC mitigation targeted at physics simulation

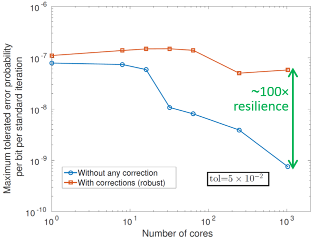
- For Sandia codes, we seek better understanding of how to anticipate, mitigate, and/or diagnose the effect of silent errors
- Basic principles aim at building efficient algorithm-based fault tolerance for SDC into physics solvers
    - Predictive simulations of physical systems must already be robust to various numerical and statistical errors/uncertainties
    - Hardware errors can be damped in analogous ways
- Our thesis: The dynamics of physical partial differential equations (PDEs) can support efficient ultralocal (within cache) detection and recovery, achieving stability to isolated occurrences of SDC
    - Handling silent errors quickly and transparently (like standard numerical errors) reduces the cost of a false positive

Building blocks can enable silent-error-tolerant solvers

- Mitigate silent data corruption when performing linear algebra operations in PDE solvers
  - Correcting bit flips in data when loaded from memory, just before use
  - May enable more efficient but "lossy" architecture co-design options



Large corruptions detected as outliers and corrected



Correction enables conjugate gradient to converge for up to 100× higher rates of emulated memory bit flips

Better design of digital systems can improve engineering

- In a traditional mathematical view, a digital system is an idealized logical machine
  - Still much room for design flaws to hide in complexity
  - Formal methods can help address this problem
- In a systems engineering view, a digital system is a design abstraction used for flexibly relating one physical system (computing device) with another (outside world)
  - This introduces the additional complications of cyber-physical systems and out-of-nominal behavior
  - Extending formal methods, including via complex systems theory, can address these broader concerns
  - National security applications can benefit from stronger analytic understanding of digital system behavior