SAND2015-2796C

Digital System Robustness via Design Constraints: The Lesson of Formal Methods

Jackson R. Mayo, Robert C. Armstrong, and Geoffrey C. Hulette

Sandia National Laboratories, Livermore, CA 94551, USA

{jmayo, rob, ghulett}@sandia.gov

Sandia National Laboratories is a multi-program 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.

The complexity of digital systems makes them unanalyzable in general

- Just because we designed and built a digital system, and it operates perfectly as a mathematical engine, doesn't mean we understand everything it can do
- Digital systems are an exemplar of complex systems: engineered or evolved systems that behave as large-scale information networks
- Turing's halting problem: the behavioral properties of such an information network cannot be predicted in the general case
  - Safety and security requirements (what the system must *not* do) cannot be verified by testing
  - Unforeseen vulnerabilities are routinely found in deployed hardware and software

Formal methods can prove behavioral properties of specific digital designs

- Formal methods apply automated logical reasoning to exhaustively analyze a mathematical model of a system
- To get around the halting problem, the system design must be expressible in a modeling language that is suitably *constrained* to be analyzable
- Two main kinds of formal tools exist
  - Theorem provers: proving requirements with general logical reasoning and human guidance
  - Model checkers: exhaustively checking requirements in all reachable states using reduction heuristics

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 proven 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)

Yet, systems designed using formal methods appear more robust, even beyond what is proven

- The SMACCMPilot project (Hickey et al. 2014) developed control software for a drone in the Ivory domain-specific programming language (DSL)
  - Ivory constrains against some unexpected behavior by enforcing basic memory safety properties
  - The resulting drone software was dubbed "unhackable" after extensive red-teaming
- The Compcert C compiler (Leroy 2009) was developed in the Coq theorem prover, tantamount to a restricted programming language
  - Extensive randomly generated tests ("fuzzing") uncovered hundreds of errors in mainstream C compilers but none in Compcert's core (Yang et al. 2011)

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
  - Example: the LLVM compiler infrastructure has undergone some formal analysis, but fuzzing suggests it is no more robust than other compilers

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

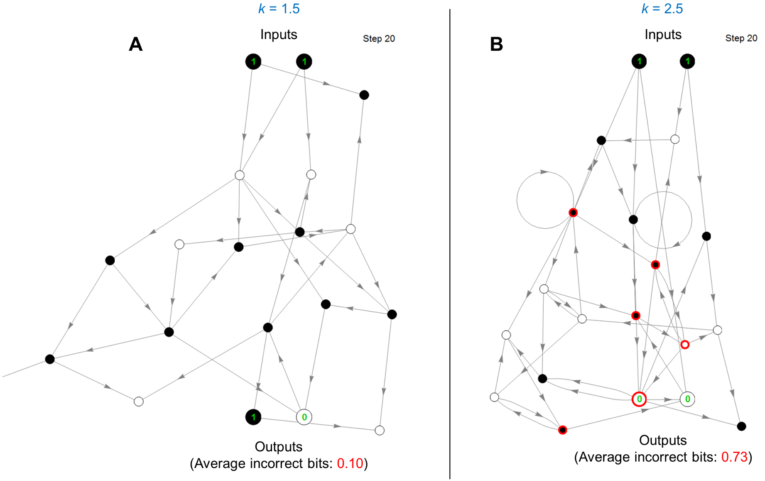
Abstract models of computation suggest approaches to improve robustness

- Theory and practice indicate that restricted models (à la DSLs) enable more powerful reasoning about behavior
  - Increasing analyzability:
    Turing machines → pushdown automata → finite-state machines
- Whereas the conventional model (Turing machine) is "uniform" (algorithm independent of data size), "non-uniform" models with bounded capacity are both more tractable to *formal methods* and more prototypical of *adaptive systems*
- To concretely explore the potential for robustness from non-uniform computation, we consider an idealized programming model with adjustable stability properties: Boolean networks (Glass & Kauffman 1973)

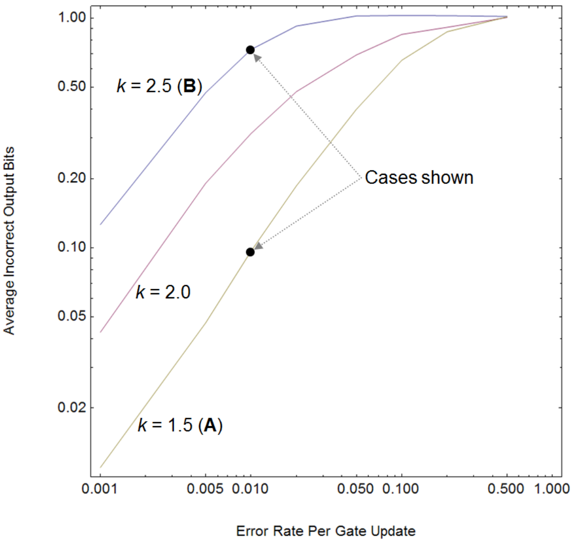
Boolean networks provide a simple representation of digital logic

- 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

Summary: Formal modeling appears to constrain digital designs in ways that increase robustness

- Requiring subcriticality is a constraint that generally makes a digital design more difficult to create but confers valuable predictability on behavior
- This is analogous to what is seen in the more specific approach of formal methods: formally informed designs exhibit robustness well beyond what is directly proven
- Boolean networks provide an idealized setting in which robustness benefits can be quantified
- We look forward to more data regarding real-world results of formally informed design of complex digital systems