

# **Leveraging Formal Methods and Confronting Complexity for Digital System Reliability and Security**

**Sandia National Laboratories, Livermore, California**

Sandia National Laboratories is a multiprogram laboratory operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.



# About Sandia

---

- A Department of Energy national laboratory
  - Primary mission is to develop, engineer, and test the non-nuclear components of nuclear weapons
  - R&D in arms control, nonproliferation, waste disposal
  - Research in computer science and supercomputing, computational biology, mathematics, alternative energy
- Located in Albuquerque, NM and in Livermore, CA



# Our digital systems

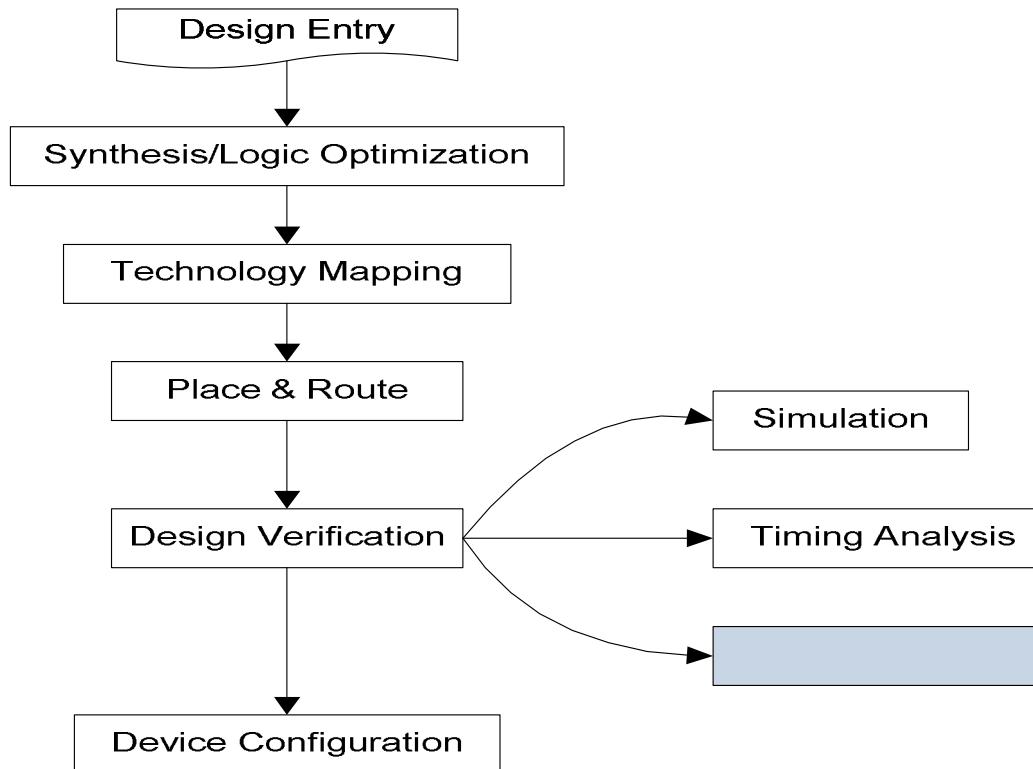
---

- Custom digital systems based on FPGA/ASIC
- Embedded systems for high reliability
- Command and control networks
- Software for simulation



# Our current design flow

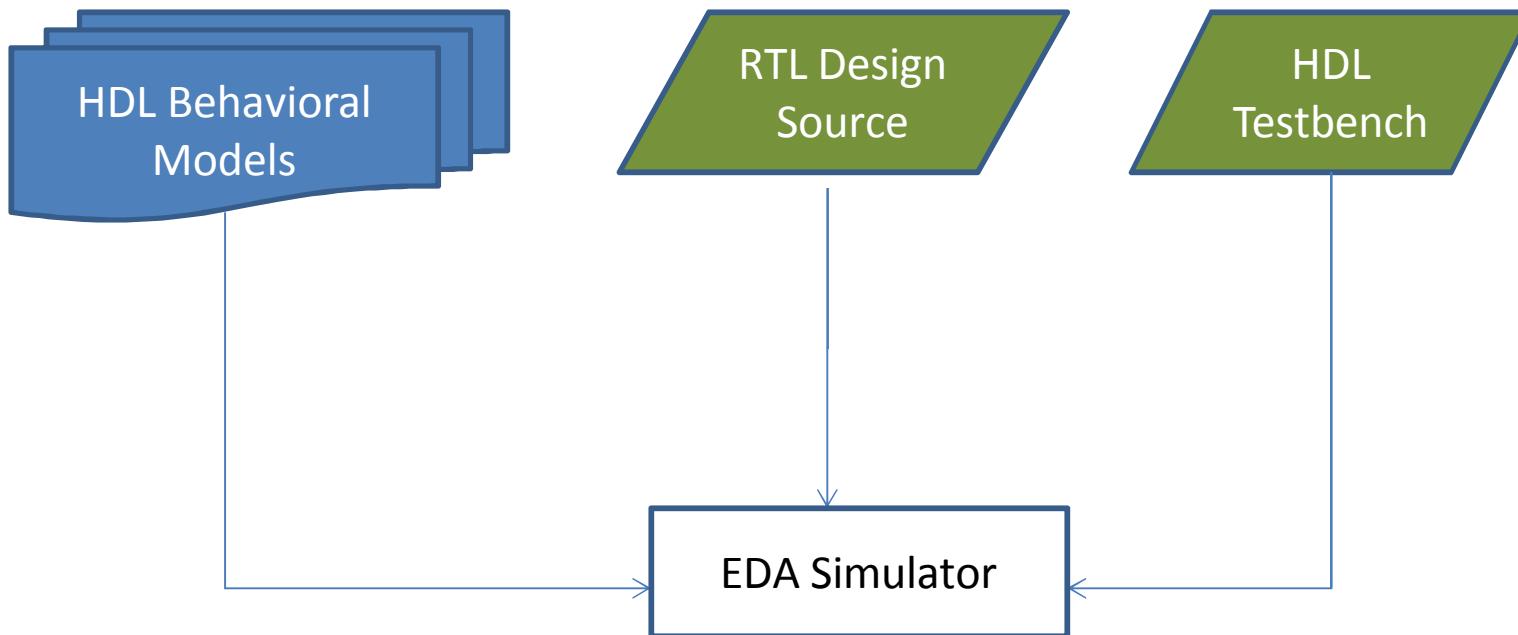
---





# Behavioral simulation flow

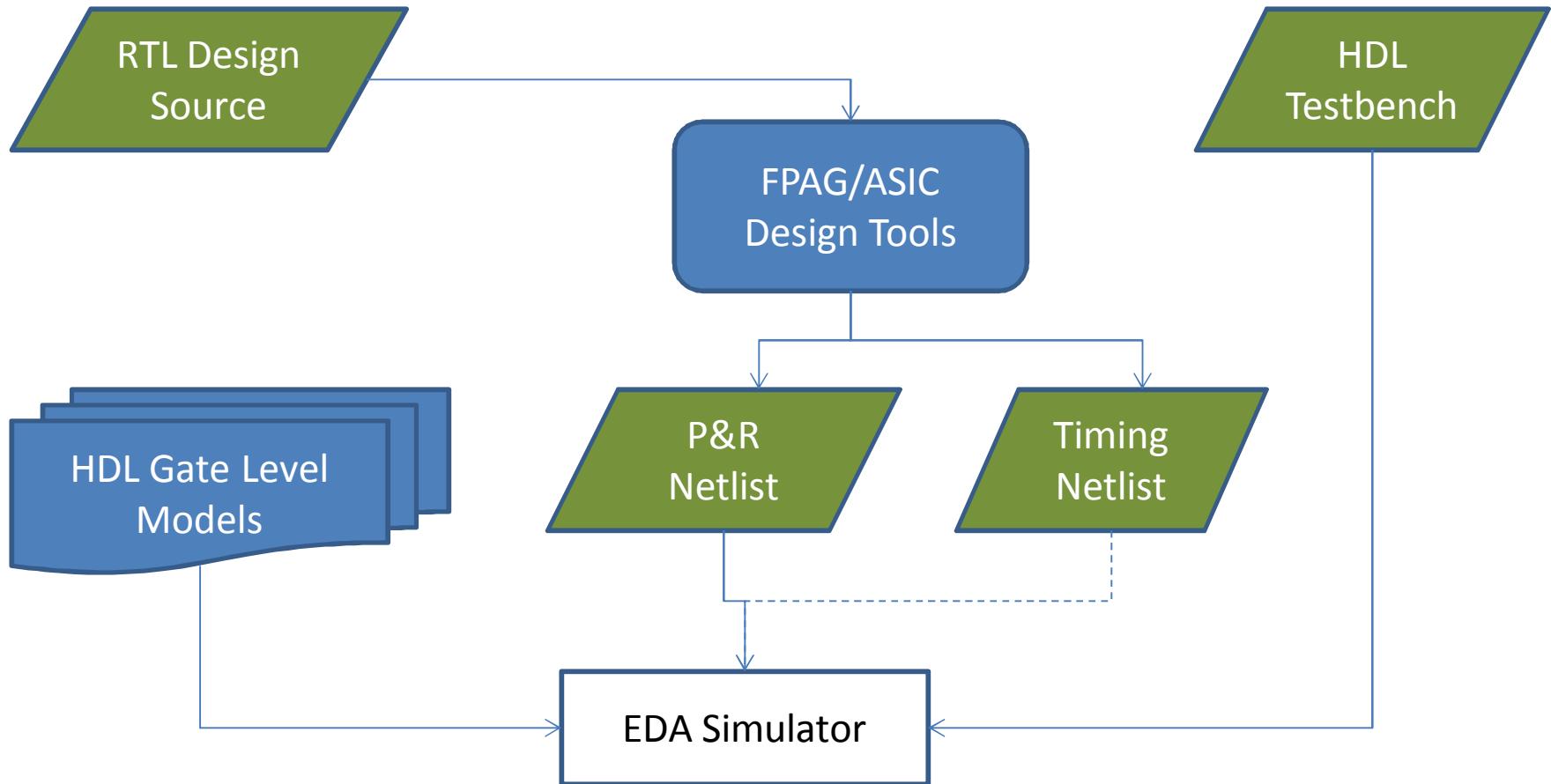
---





# Gate level simulation flow

---





## Our goal

---

- Exploring effective formal verification methodologies for customized digital systems
- Exploring applicable formal methods for complex systems related to cybersecurity

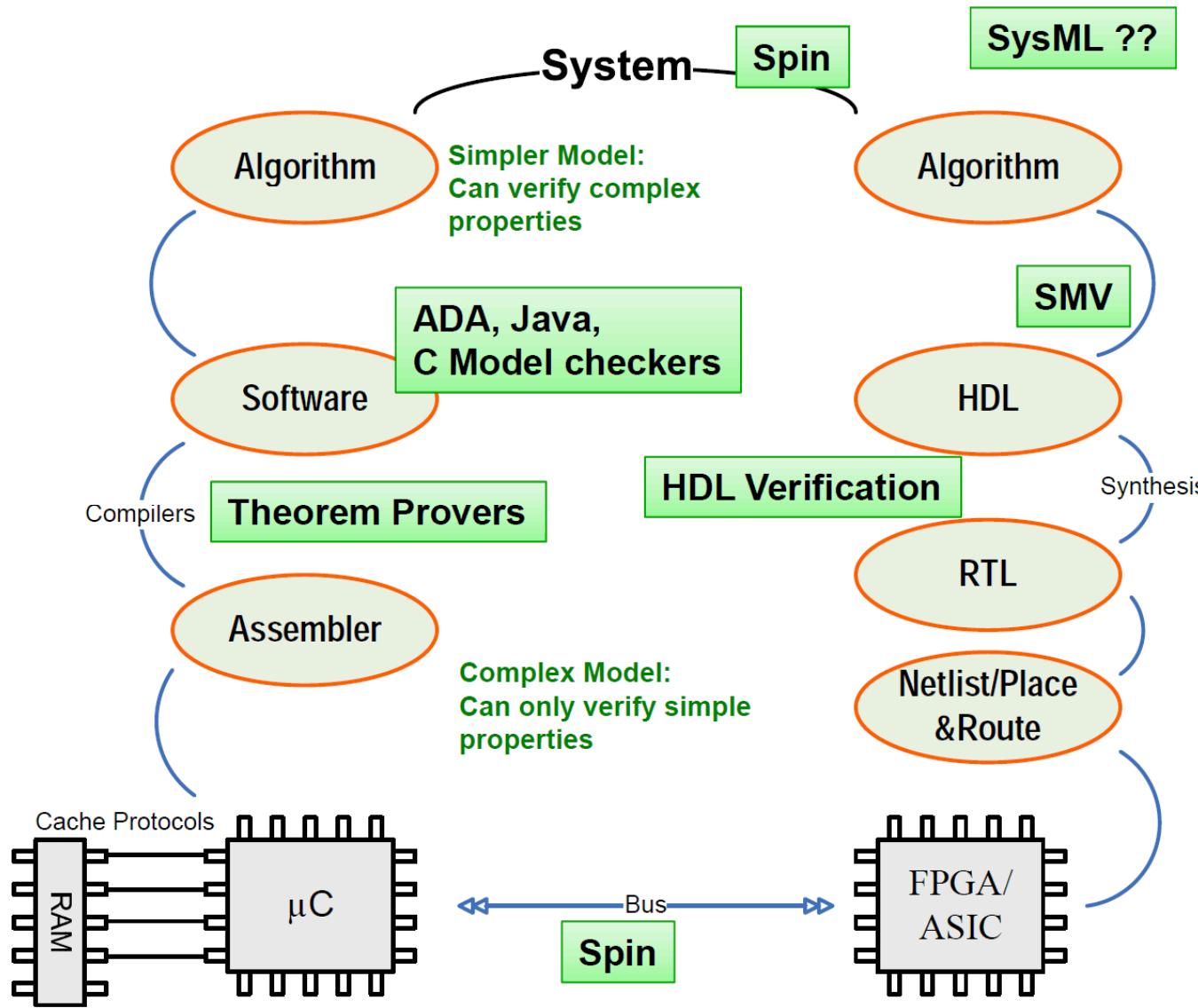


# We view formal verification as one facet of the digital complexity problem

---

- Includes issues in cybersecurity, reliability, and safety
- Complexity causes digital systems to have unknowable and, in general, unanalyzable faults/vulnerabilities
- Formal methods are currently relegated to simple systems or high levels of abstraction
  - New approaches are improving on this

# Formal methods can be applied at various levels of abstraction





# Our challenge

---

- Highly complex systems compared to most industrial applications
- General FV algorithms do not cover the entire space for our systems
- Specific properties related to extreme environments are hard to model and verify



# Commercial formal verification tools

---

Company	Equivalence Checker	Model Checker	Simulation Based Verification
Cadence	Encounter Conformal Equivalence Checker	Incisive Formal Verifier	Incisive Enterprise Verifier
Jasper	N/A	JasperCore	N/A
Mentor Graphics	FormalPro	Questa Formal	Questa Core/Prime (ModelSim)
Synopsys	Formality	Magellan	VCS



# Open source tools

---

- Cadence SMV model checker
- Distributed Real-time Embedded Analysis Methods (DREAM)
- Java PathFinder (JPF)
- NuSMV
- Spin
- VIS

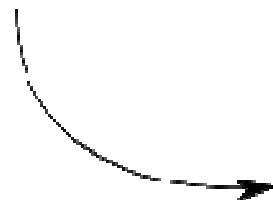


# Securing an arbitrary code is not just hard; it's impossible

---

- Restated: Generic code has vulnerabilities that are unprovable and unknowable
  - Not statistical, even in principle
  - Turing completeness demands that a generic code is undecidable

Program



vulnerabilities

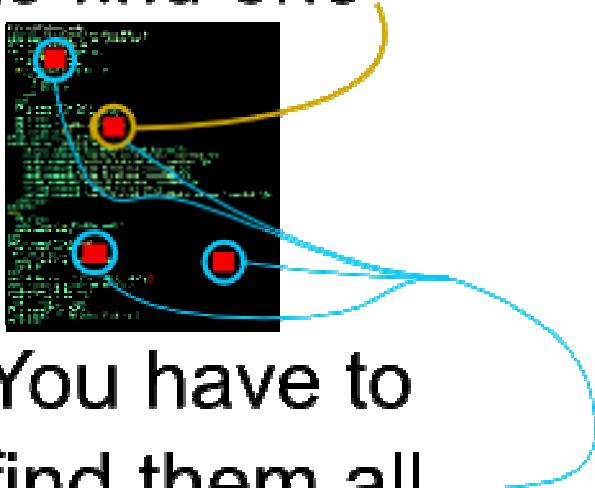
- So now what?



# Complexity makes cyber threats asymmetric

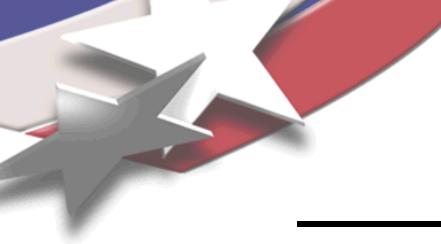
---

Bad Guy needs  
to find one



You have to  
find them all

- **Developer, user, and attacker** all don't know where the vulnerabilities are (*undecidable*)
- Worse, attacker may have planted a vulnerability
- **Asymmetry: One vulnerability compromises the whole code**
  - Developer has to find all of them (impossible in general)
- **No one can guarantee “this code is clean” or even quantify improvement**

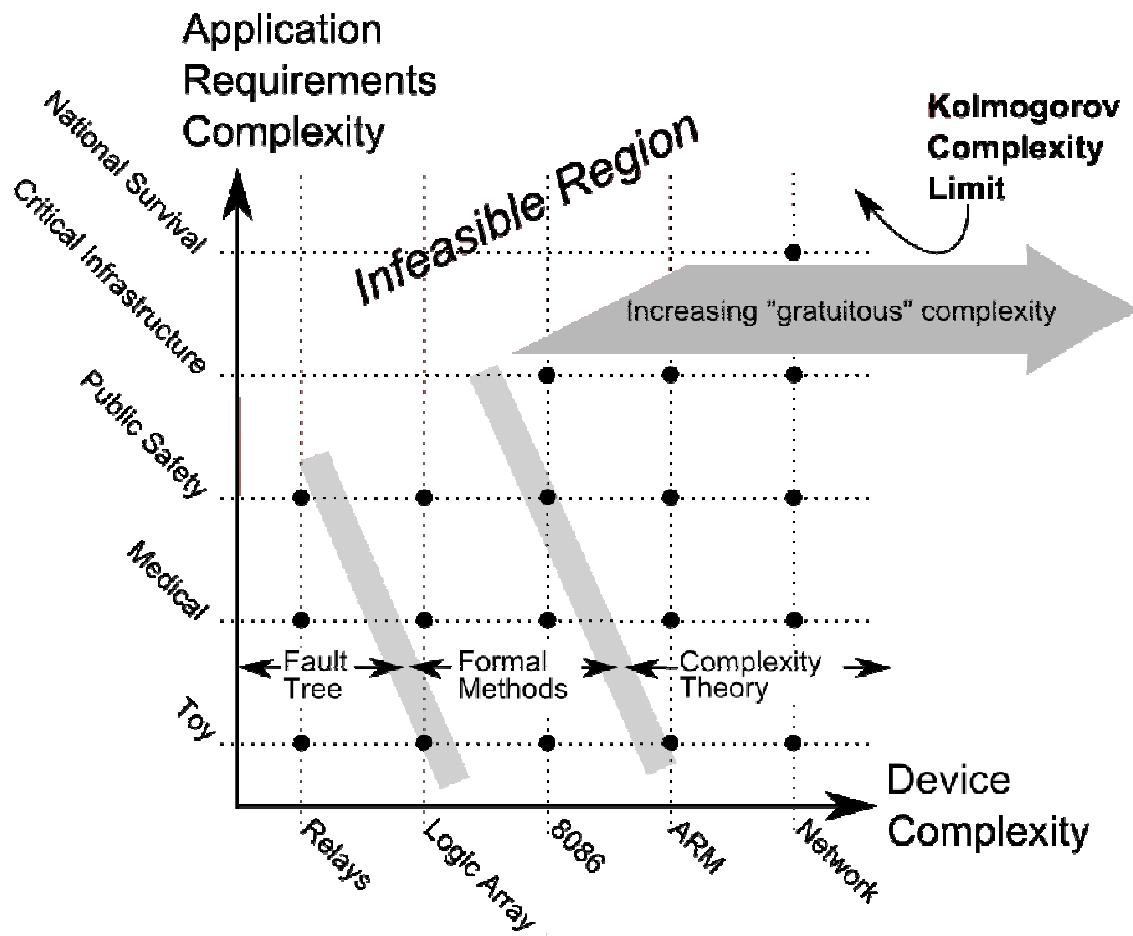


# What is complexity?

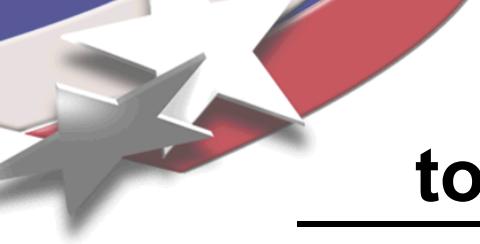
---

- Complex systems are characterized by **large** numbers of interacting entities where even a **few** entities can **strongly** affect system behavior
- Complex systems are **irreducible**; their behavior is **emergent** and not evident *a priori*, but is accessible via **observation** and **simulation**
- Examples are **ubiquitous**
  - Living things and ecosystems
  - Human societies, economies, and institutions
  - Highly engineered artifacts – e.g., airplanes, NWs
  - Large-scale infrastructure – e.g., power grids
  - Computer software, hardware, and networks

# Complexity space illustrates tradeoffs in device engineering and analysis



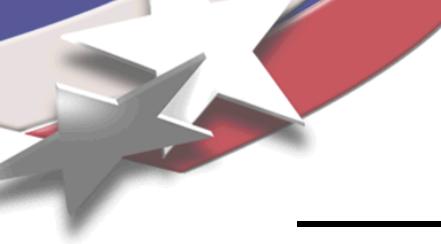
- **Pink region is what cannot be built**
- **Rest of plot shows how analysis can be done**



# Formal methods are a bridge to complexity, filling an important gap

---

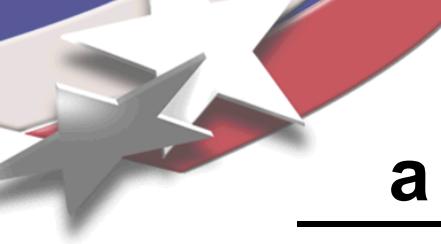
- Formal methods use computer analysis to verify digital systems **rigorously and exhaustively**
  - Applicable to less complex systems that are still beyond the reach of manual analysis
  - Widely used in high-consequence industrial applications
- Verification of components does not generally translate to verification of whole system
- Irreducible complexity enters when exploring entire state space is infeasible
  - Reliability and security assertions become **probabilistic**
- Both formal verification and complexity science are vital for gaining confidence in digital systems



# Complexity science offers a new perspective on modeling and design

---

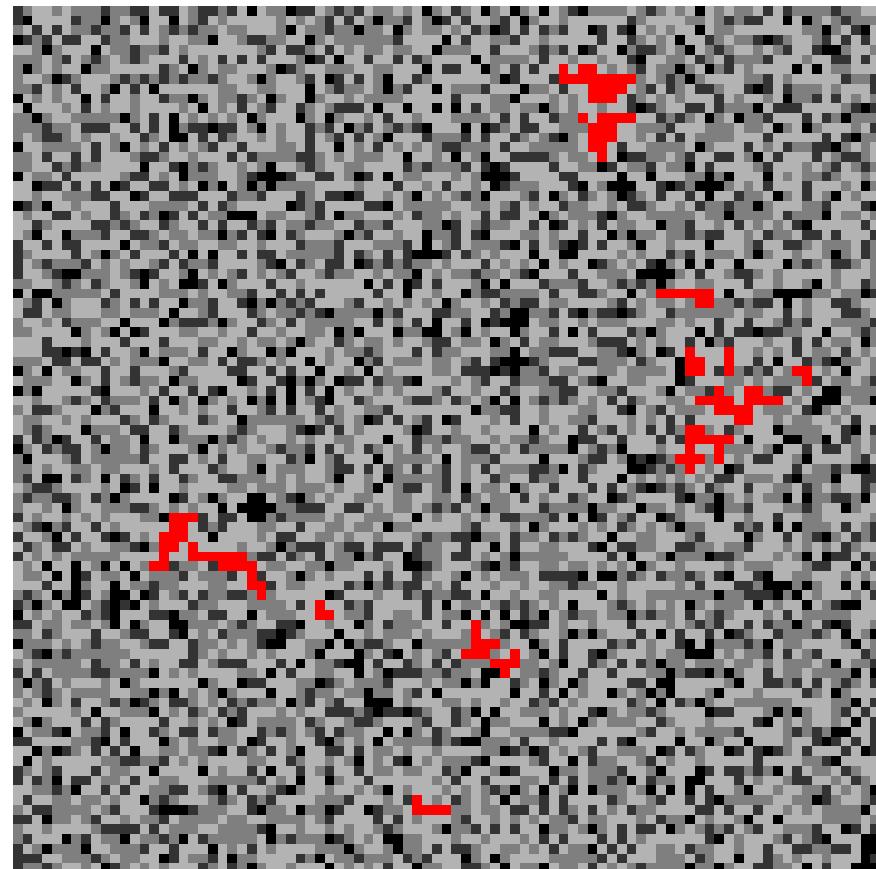
- Most real-world systems are too intricate to analyze directly; they are **irreducible**
- Reductionism requires “bottom-up” understanding
  - Use expert knowledge to model component entities
  - Validate system model vs. observations
  - Make each component entity as reliable as possible
  - Formal methods are the pinnacle of this approach
- Complexity science provides “top-down” insight relating system structure to emergent behavior
  - New modeling paradigm: Identify entities by abstraction from idealized models with known emergent behavior
  - New design paradigm: Build real systems based on models with desired emergent behavior



# Self-organized criticality is a simple example of emergent behavior

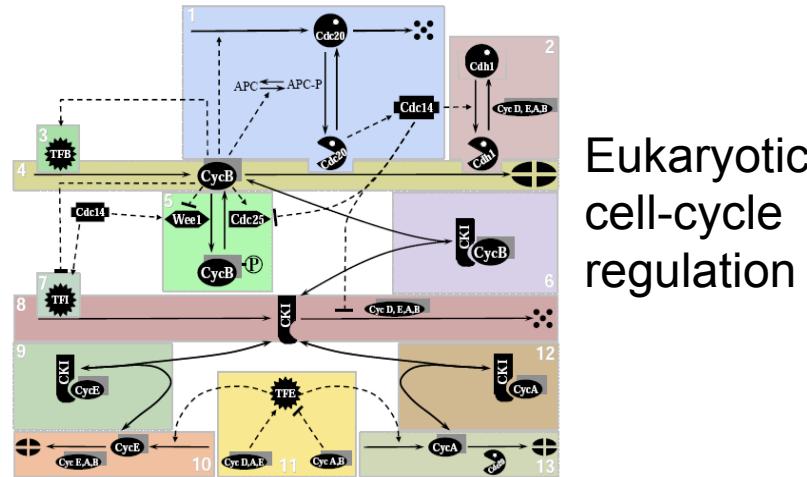
---

- “Sandbot”: cyber model of coordinated malware
- **SOC is spontaneous development of multi-scale phenomena with power-law distributions**
  - Similar to thermodynamic criticality but without tuning
- **Illustrated by sandpile model: physics-like cellular automaton**
  - Sand is sprinkled randomly
  - Avalanches occur at all scales



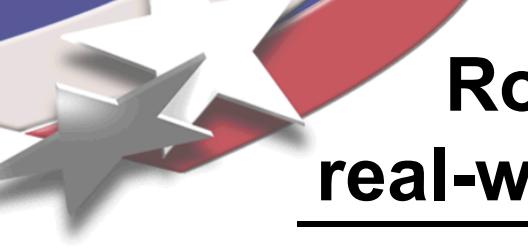
# Complexity is a fact of “life”

- Biological phenomena are a prototype and inspiration for many complex domains
  - Life involves a large chemical regulatory network



Eukaryotic  
cell-cycle  
regulation

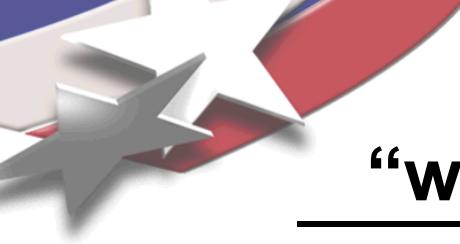
- “Game of Life” model is based on population dynamics
- Bio concepts pervade computing (viruses, mutations)
- **Biology typifies complex couplings of manmade systems – economy, energy, cybersecurity**



# Robustness is key to understanding real-world systems with “organic” behavior

---

- Highly optimized tolerance (**HOT**): Systems *designed* or *selected* to perform well despite perturbations
- Robustness is necessary for biological evolution and for effective engineering
- HOT systems exhibit power-law distributions like SOC but have organic structure (not self-similar)
- Adapted robustness to one set of perturbations induces **extra fragility** to different perturbations
- Indeed, rare but catastrophic failures are seen in highly engineered/evolved systems
  - Electrical blackouts, cyber shutdown of Estonia, financial panics, hacker penetration of bank database, etc.



# Complexity can address “whole system” robustness and stability

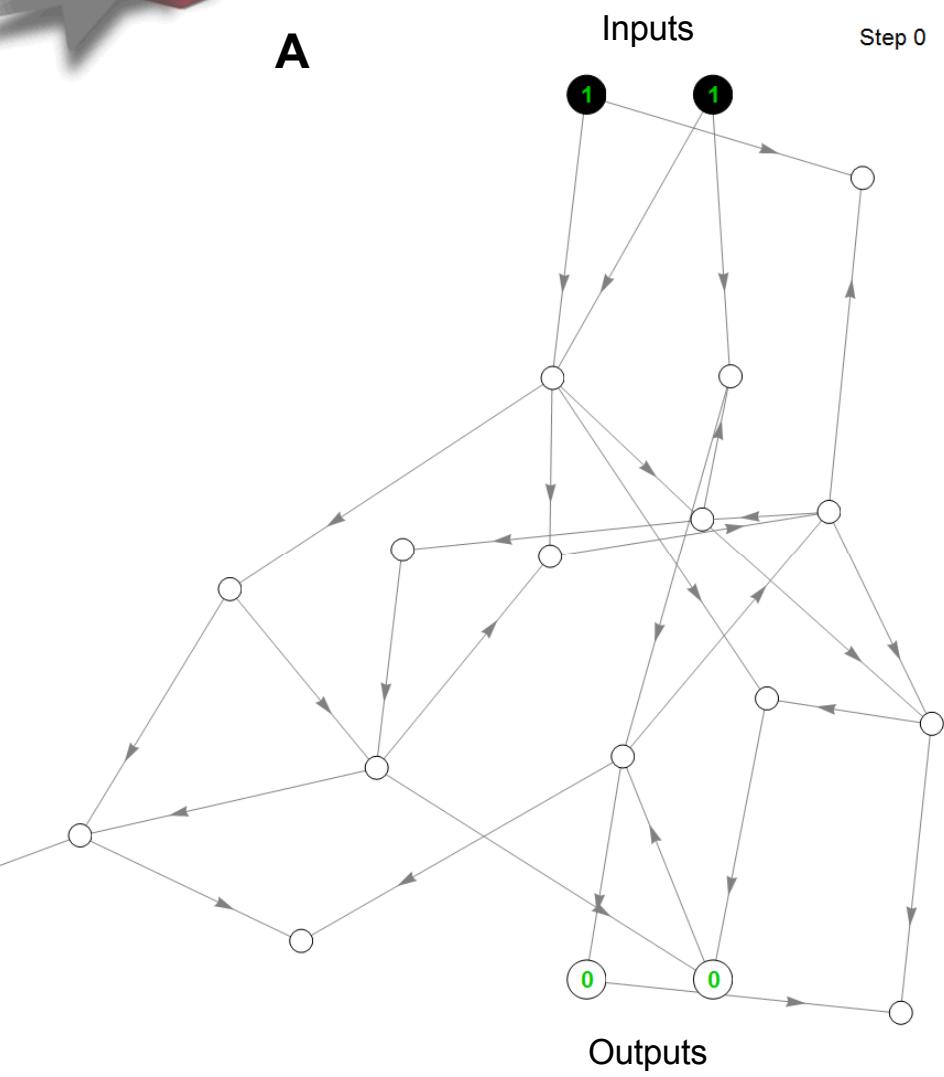
---

- Consider designing a digital circuit to add two 1-bit numbers (a “half adder”)
  - This is among the most basic functions appearing in microelectronics
- There are many ways of composing logic gates to implement this functionality
- The next slide shows two such circuits; each performs as a half adder when run for twenty steps
  - Shown correctly adding  $1 + 1$  to get the binary result  $10$
  - They also give correct answers for the other possible inputs

**A**

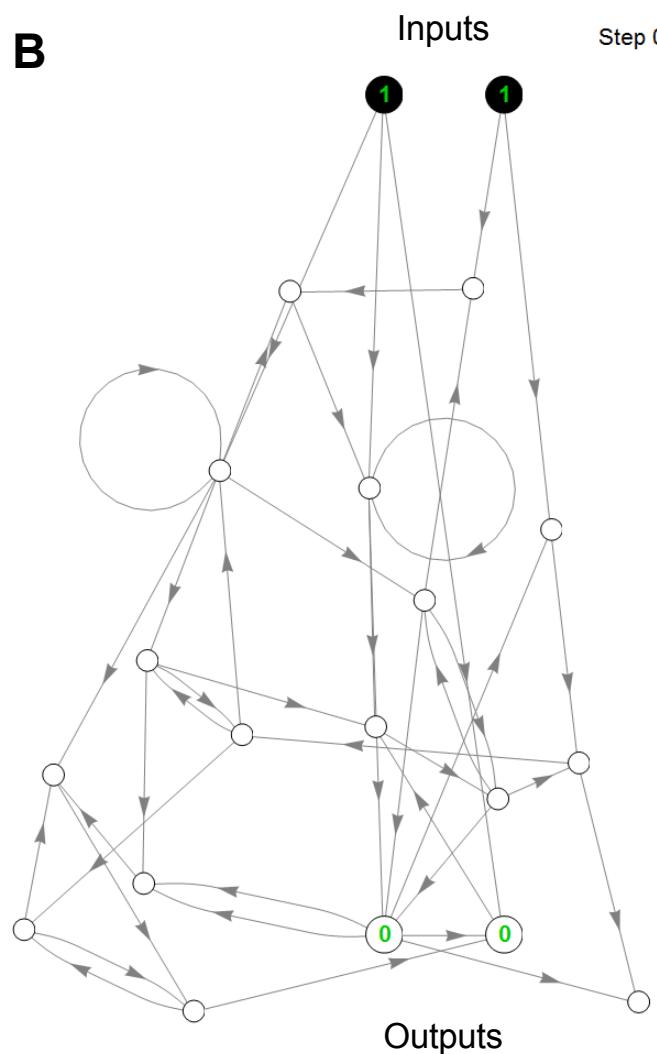
Inputs

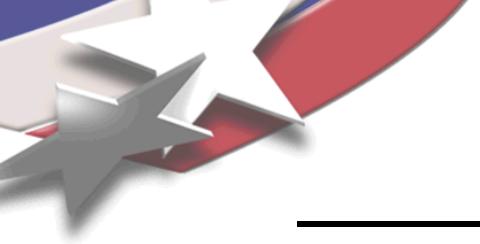
Step 0

**B**

Inputs

Step 0

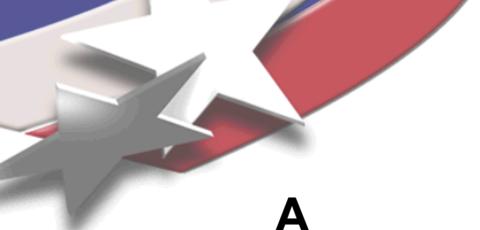




# What distinguishes the two implementations? *Resilience*

---

- For this very simple functionality, both circuits can be verified by exhaustive testing
- More realistic circuits cannot be tested exhaustively, so we need to understand the effect of untested states
- In this example, we introduce occasional gate errors to represent unanticipated behavior
- The next slide shows a typical run of each circuit with a 1% error rate per gate update
  - States that deviate from the ideal run are outlined in red
- Circuit A has much less error in the final output (greater resilience) than circuit B – why?
  - In this case, average inputs per node (*k*) makes the difference

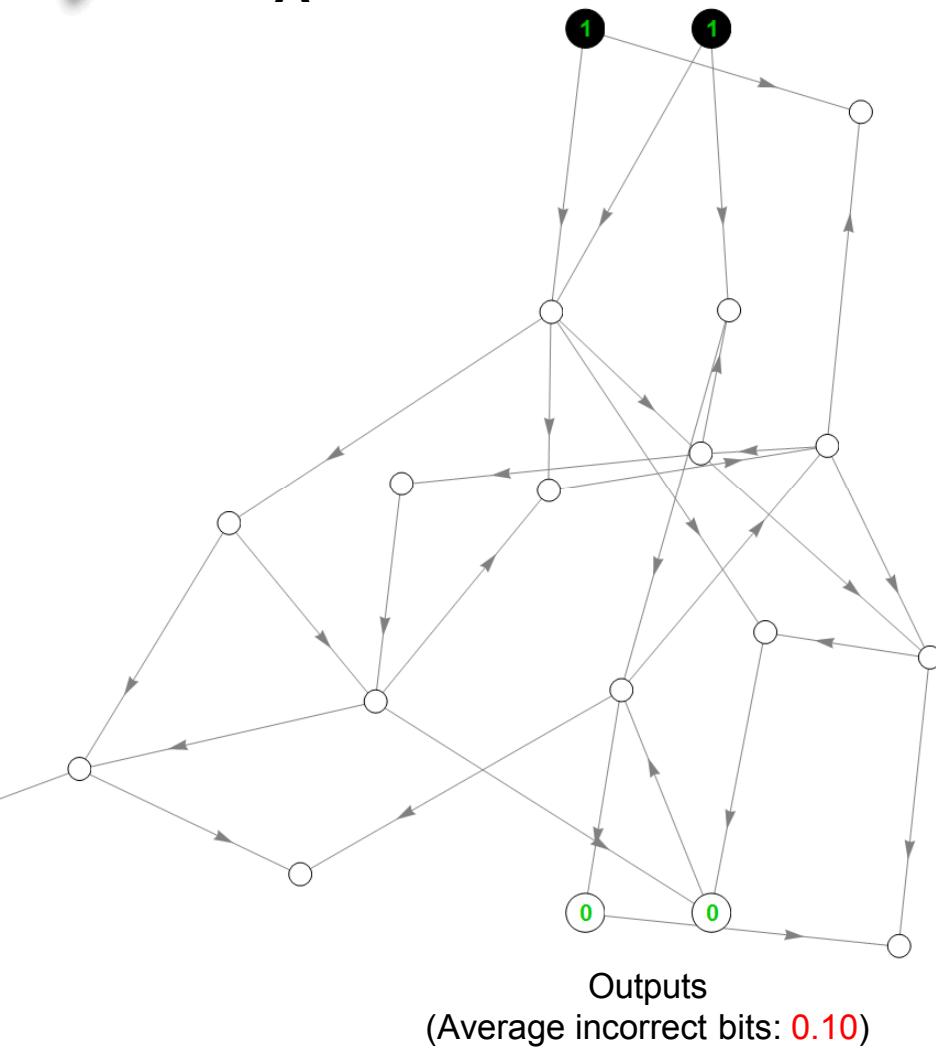


$k = 1.5$

**A**

Inputs

Step 0

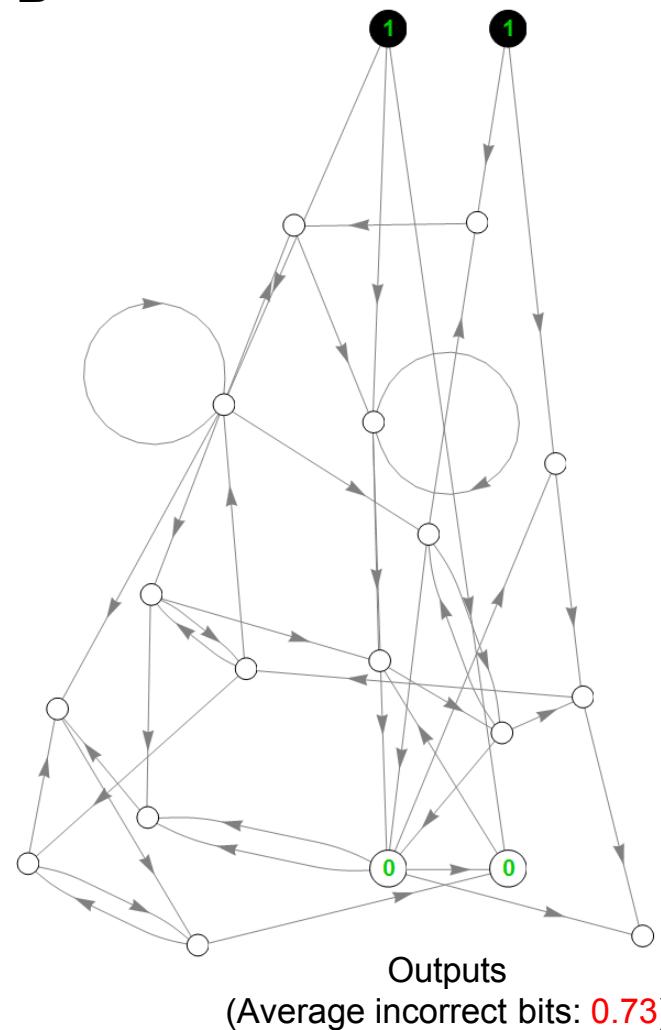


$k = 2.5$

**B**

Inputs

Step 0



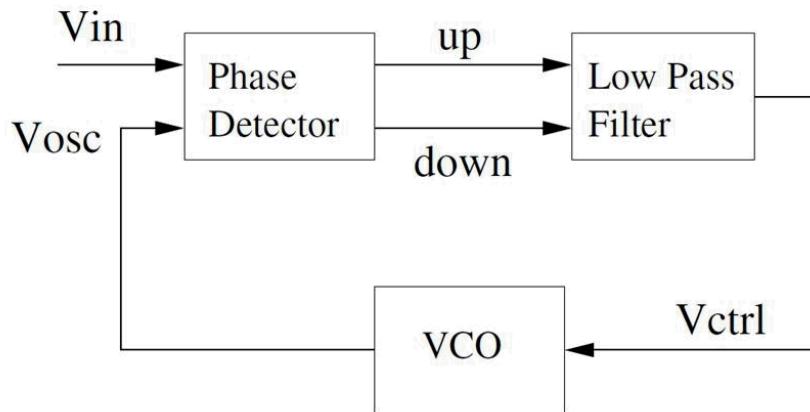


# Analog mixed signal control systems

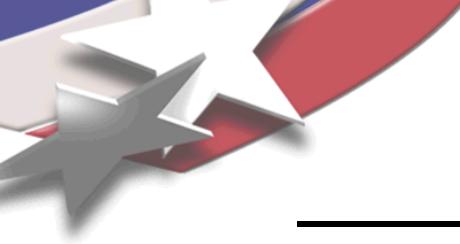
---

**Hybrid systems:** Dynamical systems that exhibit both discrete and continuous change

- Their discrete variables are updated in discrete steps that consume no time (resulting in jumps)
- Their continuous real-valued variables (clocks and drifting clocks) are updated as continuous functions while time elapses (during delays, resulting in so-called “flows”)



PLL diagram (Little et al, 2004)

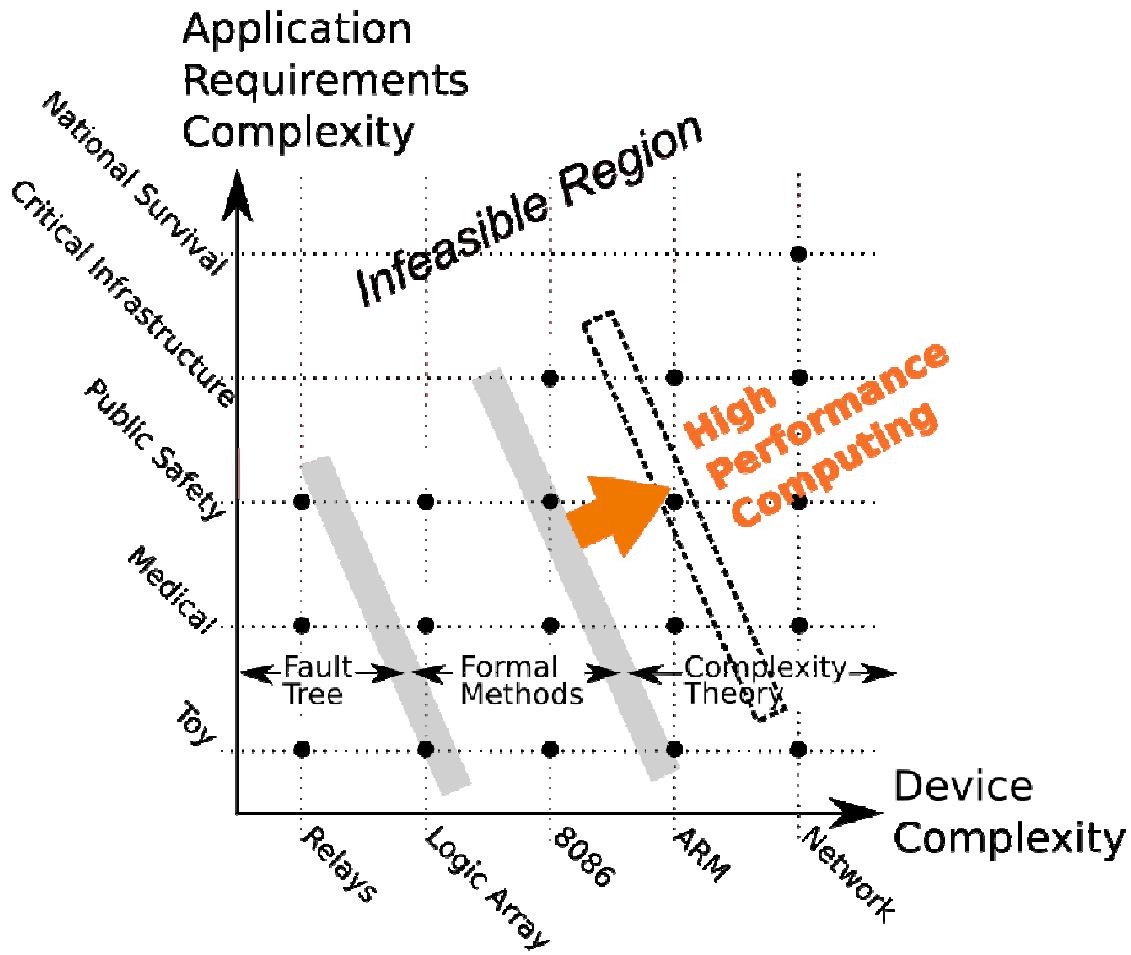


# Today formal verification is limited to the digital domain

---

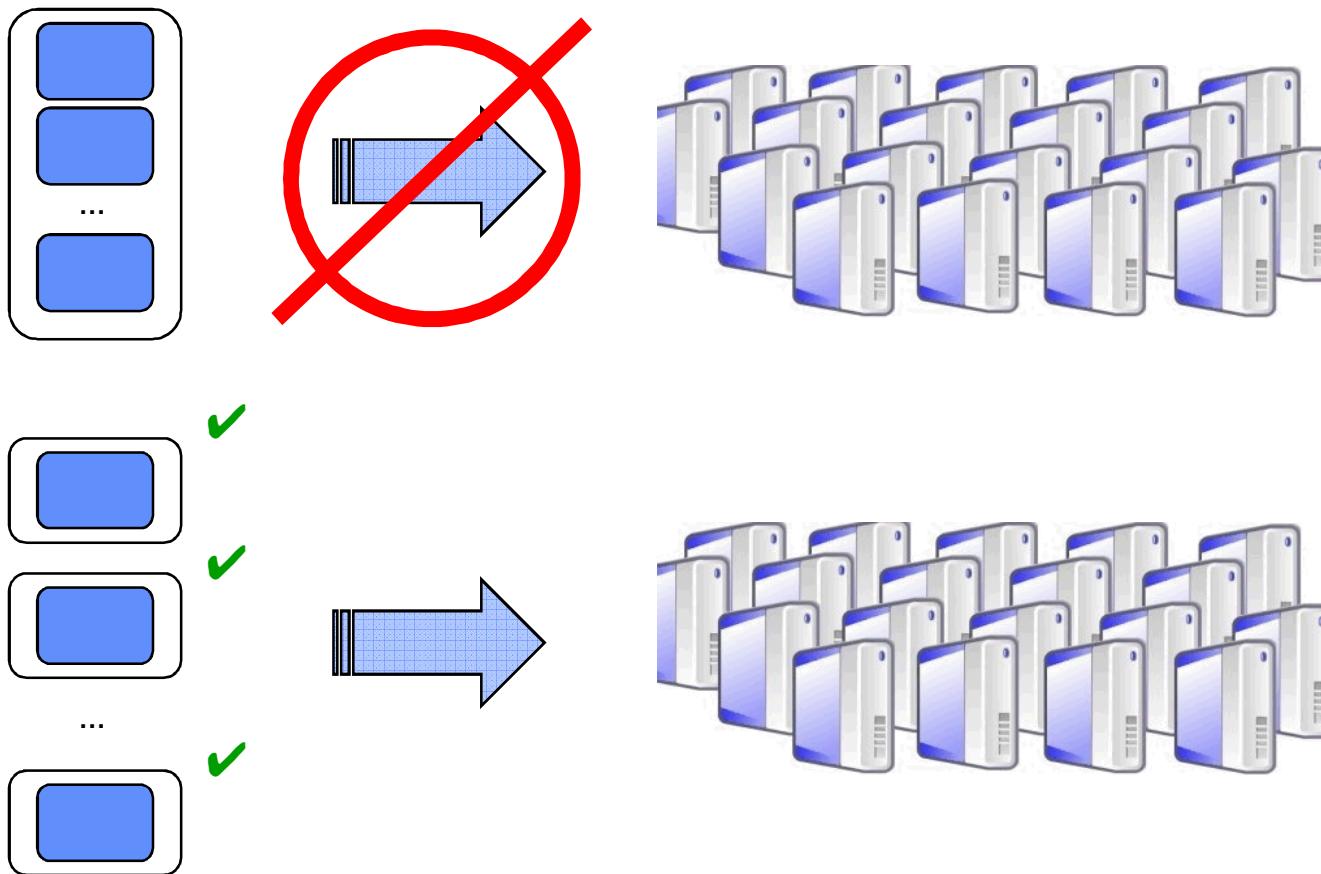
- **A computer model is generated for the digital logic**
  - Very accurate as long as:
    - The circuit behaves digitally
    - Inputs remain in their expected range
- **Not considered in formal verification:**
  - Digital circuit ceases to behave digitally because of extreme environments
  - Out-of-nominal digitized analog inputs appear from the external environment
- **We seek to broaden current techniques to verify function for digital systems in extreme environments**

# Digital system analysis research will increase utility of HPC in simulation



- Complex system simulation leverages HPC but confronts issues of tractability and V&V that current research is addressing
- HPC, formal methods, and complexity theory can work together to expand our capabilities

# Composition of formally verified systems



Decomposing a complex digital design may enable component-level formal verification using HPC

Composition to the whole device is nontrivial, but the formal results can usefully constrain behavior