SAND2012-4402P

More-reliable Methods of Verification for FPGA-based Digital Systems

Ensuring that critical control systems will perform as expected

A fundamental question for Sandia high-consequence electronics involves the reliability of field programmable gate array (FPGA)-based control systems. Many critical-system electronic components are being replaced by modern digital devices, thus dramatically increasing system complexity. Such systems include those used in aerospace applications, as well as those upgraded as part of nuclear weapons life-extension programs, and the updated designs commonly rely on FPGAs to implement sophisticated logic. A key outcome of this trend is the need to validate the performance of such digital systems to the greatest extent possible (fig. 1).

Figure 1. Searching immense volumes of code for faults/vulnerabilities can be a daunting proposition.

FPGAs

Ubiquitous in modern hardware, FPGAs are electronic components that are somewhat akin to a tabula rasa, a blank page. Perhaps a more appropriate analogy would be a child's play tool for improving reading—a magnetic slate filled with movable words that the child can reconfigure into a variety of meaningful sentences. In FPGAs, the words are logic blocks that can be connected together in a variety of configurations determined by a user who is addressing specific hardware design requirements. Based upon tools provided by the vendor, FPGAs are thus designed to be configured by a customer after manufacturing, that is, after their logic functional blocks are implemented. One advantage of FPGAs over custom hardware is that they are harder to attack because the design for a processing functionality is not pre-loaded onto a device. Among their other uses, FPGA characteristics make them suitable for implementing cryptographic applications. However, while they offer this attack resistance and flexibility, they can also comprise yet another source of problematic issues, in that, in using the vendor's tools to configure them, a hardware engineer may introduce unintended vulnerabilities (fig 2).

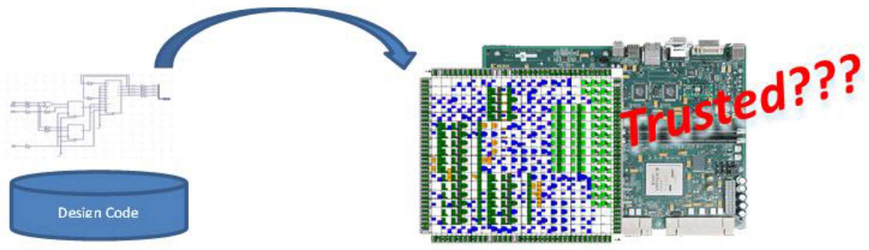

Figure 2. The process of using vendor tools on design code to pattern the logic elements of FPGAs for critical systems may introduce unanticipated vulnerabilities that simulation would not necessarily account for.

SIMULATION

Typically, validation of such systems has relied primarily on simulation, a technique that can inform the hardware designer whether the system can indeed perform as intended, that is, whether the design's intent is preserved and reflected in its implementation and performance. As a validation tool, simulation, has its limitations, however. First, for stateful systems—that is systems with a large number of possible states—run times for simulations can become prohibitively long. For example, using a state-of-the-art simulator, the time required to simulate all possible combinations of inputs (each input can take a value of 0 or 1) for an n-bit adder is a matter of microseconds for n = 4, and still less than 100 milliseconds (one-tenth of a second) for n = 8. But the simulation run time jumps to over one hour for n = 16, and for n = 32 becomes a remarkable *585,000 years* (fig. 3). The type of stateful systems involved in much of Sandia's mission space are often prohibitively difficult to simulate. And there is yet an even more crucial issue with simulation, namely that it can be somewhat blind to bugs, able to discover them only if clever test cases are used as the basis for guiding which system properties to simulate. There is no guarantee that disastrous bugs will all be discovered. While simulating a system can inform designers that it does what was intended in its design, it cannot necessarily also verify that it does not do what was not intended. In other words, simulation is generally not good at catching all unintended operational states—bugs—in a design.

n	Simulation run time
1	4 μs
2	16 μs
4	256 μs
8	65 ms
16	1.2 hr
32	584,942 yr

Figure 3. Comparative run times for "complete" simulations of an n-bit adder.

In over 10 years working in the semiconductor industry for a leading FPGA company, Yalin Hu was repeatedly faced with this validation dilemma. As she poses it: "Through the years, I have seen customers facing the same question again and again: 'how do I know . . . how confident can I be . . . that what I want is what I get"?

Upon her arrival at Sandia, Yalin noted that there were critical systems caught in this limbo, whereby simulation was the primary tool being employed for verification, even while other validation and verification methodologies were beginning to enter the broader realm of FPGA-based system analysis. Using the vehicle of an early career LDRD research award, Yalin has been adopting and

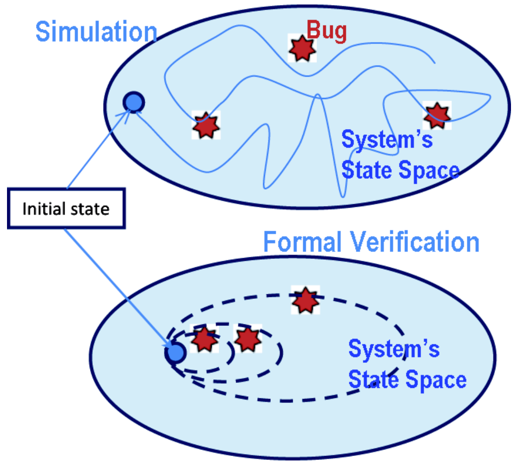

Figure 4. **Simulation** uses test cases to examine a system's state space in a somewhat incoherent fashion, and the probability of bug discovery is highly dependent on the cleverness of the test cases used to guide the simulations. Bugs are easily missed. By contrast, **formal verification** systematically searches state space regions for bugs; eventually, the FV algorithm will search the entire state space, discovering all bugs.

modifying other methods for verification of Sandia critical systems.

Formal Verification

These methods fall under the rubric of *formal verification*, a compendium of approaches employing rigorous mathematical proof that a hardware design satisfies certain specified properties. These include specific functionalities, timing properties, structural properties, and fault tolerance. Logically, this more-rigorous approach is necessary for mission-critical Sandia systems that are highly concurrent (simultaneously processing several data streams in parallel), that often operate in harsh environments (such as the high-radiation environment of outer space), and that, of course, are high-consequence systems, whose failure can have catastrophic outcomes, and whose fault tolerance is rather restricted. Formal verification is a technique to catch faults, that is, to verify that a system's array of reachable states does not include properties and states that were not intended in its design. It exhaustively explores all regions of a system's state space to uncover such incorrect system behaviors (fig. 4).

Under that formal verification umbrella are the techniques of model checking (MC) and theorem proving (TP). Each has its advantages and its drawbacks. Model checking is a computationally intensive approach in that it entails an exhaustive examination of a system's collection of reachable states (its state space) to check that desired properties hold. This is, of course, a formidable task for complex (stateful) systems because of the large number of states that must be validated. Theorem proving is a method for logical derivation of system properties that is performed by mathematically defining the system's implementation. This is far from a trivial endeavor, requiring a significant investment of human intervention to construct such a mathematical definition of the system.

In order to facilitate the application of these more-rigorous methods to Sandia critical systems, this project is studying RAM (random access memory) a quite stateful system and is, more importantly, employing a decomposition approach, that is, breaking down a high-complexity problem to several lower-complexity problems.

Decomposition

Up to this point, there has been a relative dearth of research into this decomposition methodology, and therefore, Yalin Hu's LDRD research stands to provide Sandia critical systems with a great advantage in terms of their trusted functionality. The approach can be most easily expressed as

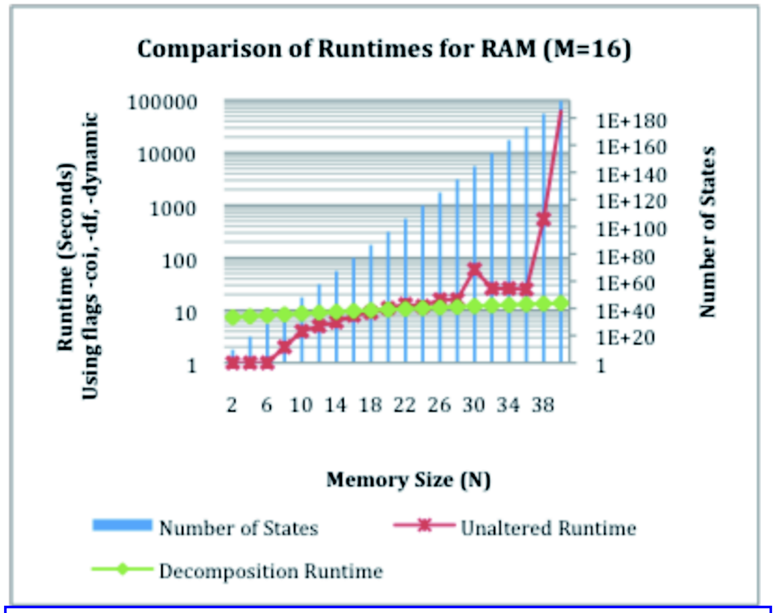

Figure 5. Comparison of runtimes for model checking of increasing RAM sizes, run as a single large problem (red) and as a set of decomposed problems (green). As the state space explodes with a problem of larger sizes, the runtime becomes prohibitive, but remains relatively stable for the decomposed problem.

decomposing a problem of size $P(N \times 2^M)$ into N problems of size $P(2^M)$. What this effectively does is to generate problems of a size that can be solved in a reasonable time period (fig. 5). Naturally, one must ultimately prove that such decomposition approaches have full validity, and part of this research is aimed at verifying that this is true. Such verification would pave the way for the use of the far more rigorous model checking with decomposition in validating critical Sandia systems.

The parallel formal verification approach of theorem proving is also a focus of this research, in that, currently, the theorems to mathematically represent a system are manually generated, consuming inordinate amounts of staff time. Were the decomposition approach successful in defining lower-complexity problems, a future goal would be to have the TP theorems be machine generated, and the hope is to develop algorithms to ultimately accomplish this. The possibility of automatizing the entire process of formal verification—model checking, theorem generation, and theorem proving —would represent an immense time savings to implement processes that would come much closer to ensuring the precise functionality of modern digital systems.

Finally, this research is not abandoning simulation, but is rather attempting to improve it. A process known as "assertion-based verification" can act as a bridge, introducing some features of formal verification into simulation to help devise simulation test cases in a more-rigorous fashion, thereby ultimately strengthening simulation as a verification tool.

It is rather obvious that within this critical systems domain, the consequences of getting it wrong, that is, of validating a system that may still, for example, possess a bug making possible a system state leading to control failure would be dire, potentially leading to loss of equipment, loss of life, and even catastrophic doomsday scenarios. These considerations boldly underline the critical nature of this research, work that should be strongly supported in every possible way.