Sandia National Laboratories

Predictive Science Academic Alliance Program (PSAAP)

Formal Methods: Improving Assurance of Cyber-Physical Systems

Karla Vanessa Morris Wright, Robert Armstrong, Jon Aytac, Noah Evans, Raheel Mahmood, Philip Johnson-Freyd, Jackson Mayo and Blake Rawlings

July 2023

U.S. DEPARTMENT OF ENERGY

National Nuclear Security Administration

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

CONTENTS

LIST OF FIGURES

1. INTRODUCTION

Nuclear weapons (NW) require a flexible, agile, and resilient life cycle as we must address an ever-changing threat environment. The NW adaptability needs are being addressed by incorporating more digital components into the systems. As a result, digital assurance must be more closely integrated with our design, implementation, qualification, and surveillance processes. The complexity introduced by the interaction of these digital components, and the digital nature of said components, demands different approaches for the verification and validation (V&V) of their design and implementation. While testing and simulation of the behaviors related to reducible systems (e.g. aerodynamics, or heat transfer) is sufficient, we cannot hope to understand digital systems by testing alone. In particular, the "always/never" requirements so essential to high-consequence systems can't be guaranteed by testing, but must be reasoned about mathematically. Traditional V&V activities must be complemented with formal methods, which provide the mathematical foundations to analyze the digital systems of interest and exhaustively verify what it is supposed to do, and especially what it is not supposed to do. Indeed, formal verification can be used anytime there are complex interactions between hardware/software and other hardware/software, such as co-design of dataflow accelerators in exascale HPC systems.

Figure 1 shows some of the different representations a digital component can take. Each representation is a level of abstraction used in the design and implementation process. To ensure the correctness of the NW systems, each successive abstraction layer must be shown to be consistent with itself and the previous layer. "always/never" guarantees may then be reasoned about at whichever layer of abstraction makes them most tractable. As the evidence package for the correctness of the component includes mathematical correspondence between each of these abstractions, proofs of behaviors in the higher-level abstraction (e.g. executable specification) are preserved by the lower (e.g. C source code).

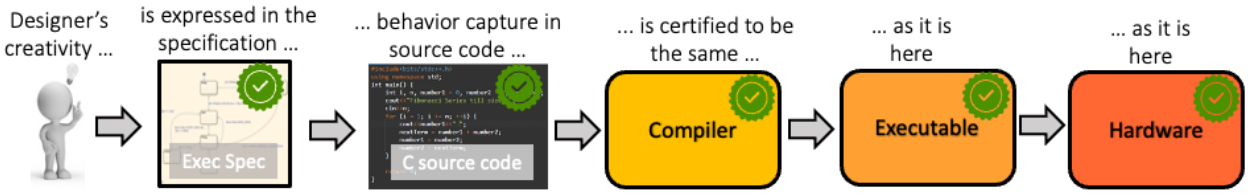

Figure 1: NW digital component representations corresponding to different abstraction levels in the digital stack

Digital assurance of NW systems necessitates not only the certification of correctness of each of its components, but also guarantees that system is correct with respect to its specification. The goal is to develop theory and capabilities needed to support a workflow that enables rigorously deriving, specifying, and analyzing NW systems and to provide evidence that their requirements are upheld throughout its digital stack. Refinement theory must be applied to check that each component design correctly implements its specification, and that the entire design will not violate any system-level requirements, including always/never requirements. The workflow should enable the construction of a fully verifiable system stack, with proven correct components, and one Q.E.D. including evidence of mathematical correspondence between each abstraction layer of the complete

design (from the high-level system design to individual component design, to implemented software and hardware designs for individual digital components). The collection of formal verification artifacts (including executable specifications, logical formulas representing formally verifiable requirements, model checking results, machine checkable proofs, etc.) will contribute to the evidence package corroborating the digital assurance case made for the qualification of the NW system and its components.

2. CURRENT CAPABILITIES

2.1. Software Formal Verification

Formal verification of firmware for NW components begins with the manual translation of a word document system specification into state machine specification by a formal verification analyst [1]. The specification, expressed as a simulatable Simulink/Stateflow diagram, may then be synthesized into our own specification language, QSpec. The specification is then translated (using our QQ compiler) into different languages to be used by several analysis tools, including NuSMV, and Frama-C. We refer to this logical specification as the "Executable Specification" because it is, among other things, simulatable.

In addition to the System specification there is a more detailed device software specification also written in Word and used to create the software implementation. This specification is also converted into Simulink/Stateflow and then to QSpec. An important part of the workflow is **proving** that the device executable specification is indeed a refinement of the System executable specification formally and mathematically. Right now, we use both NuSMV and MC3 (a Sandia-developed model checker) to accomplish this proof.

The PRT specification must be shown to be in correspondence with the C code implementation. The executable specification as QSpec is converted to the C assertion language ACSL (ANSI/ISO C Specification Language) which then can be proven against the C code using Frama-C. A certified compiler, CompCert, is used to guarantee that the binary corresponds to (refines) the C code.

Each layer of the stack is proven to the layer before to provide an unbroken chain of evidence that the implementation is everything the specification says it is.

Other approaches used in the formal verification of firmware involve the construction of a Coq proof assistant detailed functional specification for the behavior of the component. The word document requirements are formalized as properties about the functional specification, and these are proven in Coq. The C code implementing the firmware is then proven to be in correspondence with the functional specification using the Verified Software Toolchain (VST), a logic embedded into Coq which itself is proven sound against CompCert's specification of the C language.

2.2. Hardware Formal Verification

Because the certification requirements are so rigorous, System On a Chip (SOC) are fabricated in a special-purpose foundry, usually at a fairly coarse feature size for radiation resistance, limiting the SOC to a fairly low clock speed. Full processor specifications for the system AXI bus and RISC-V (next-gen processor) implement refinement by showing that for all possible behaviors the specification (written in Verilog) matches the implementation. Much of the heavy lifting comes from making sure that these specifications are complete and free of bugs.

Other peripherals related to the function of the controller are specified by designers and require an abstraction/refinement stack like the software formal verification that is yet unwritten. In the interim, the handwritten Systems specification is translated by hand directly to System Verilog Assertions (SVA), finite state machines that can be checked against the synthesizable design using model checking or constrained random testing to prove properties about the system.

3. RESEARCH AND DEVELOPMENT

It cannot be over-emphasized that the verification of the digital design relies heavily on the lucidity and the transparency of the design itself. The exact design morphology and approach must accommodate the verification approach. An arbitrary verification technique on an arbitrary design is doomed to failure (formally: is undecidable). Design constraints/guidelines that can be formalized and provided to the implementation teams are of interest as they should contribute to the creation of a successful design that can be successfully verified.

In general, tools, toolchains, and methodologies are sought to support ergonomic constructions and verification of every representation at every level of the design process (e.g., systems level, component specification level, component implementation level, software implementation level, hardware implementation level). Even though the control systems in question are fairly small by industrial standards, they are large when considering the very large and strict requirements space that NW endemic to NW.

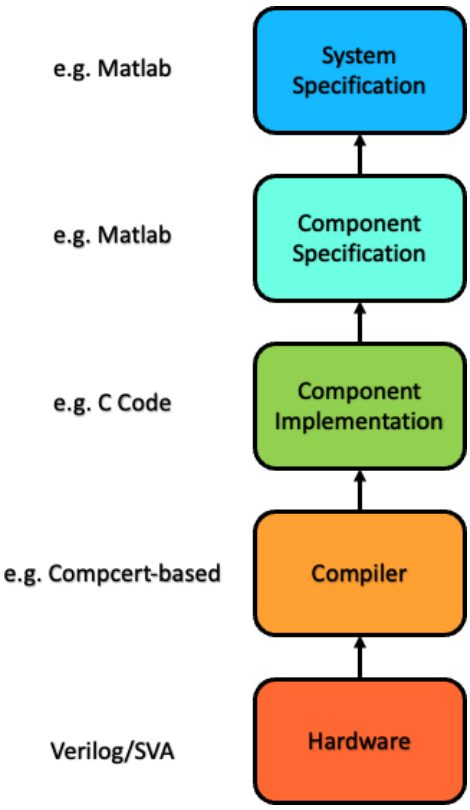

Figure 2: Layers in a typical digital controller design. Each abstraction layer is proven consistent with the layer above it. The arrows point in the direction of proof obligation of consistency.

3.1. Research Directions

- **Novel theories of abstraction/refinement** in support of scalability and workability of formal co-design. Novel approaches to modeling trace-inclusion would be welcome, but there are security and reliability properties requiring still more powerful notions of observational equivalence and refinement. To these ends, we will need new theories of abstract interpretation, new program logics, and new modes of compositional reasoning.
- Methodology and tools are required to **ensure that the overall System Specification is self-consistent**. For a complex system, like those we have in the NW space, it is harder than not to create a specification free from deadlocks and race conditions.
- Methodology and tools are required to **ensure Component Specifications are consistent with the overall System Specification**. Ideally these would employ common tools that engineers already use to create specifications and would constrain engineers (with as little pain as possible) to derive only Component Specifications that are consistent with System Specification.
- Methodology and tools are required to **prove Component-level Specifications to implementation code**. This can include partial solutions that involve SMT provers, proof assistants, etc., automated methods for deriving simulation maps, and generalized proof systems that utilize any of these, or none of these.
- Methodologies and tools that enable **Code synthesis of Component Specifications** to implementations. Even at this, the generated implementation code is unlikely to stand into the final product, since alterations necessitated by performance, size, and other constraints are inevitable. Thus any proposed method must accommodate proof repair in the altered code.
- **The Compiler is singled-out a separate entity, but in fact is several separate stages, all of which want verification**. While the Compcert compiler is formally proven to the back-end ISA, there is no corresponding guarantees for the linker, preprocessor, or libc.
- Because of scalability concerns, **automated proof systems that provide a proof certificate/ counter examples** will help the proof case for NW immensely. Much of the motivation for doing formal analysis in the first place is to provide evidence that requirements have been met, proof certificates are exactly that. Counterexamples provide a strong hint to the practitioner of what is wrong with their design, or model.
- The advent of quantum computers mean we can't just reuse the cryptography from previous designs. Work that advances the art of **verifying "post quantum" cryptography algorithms** is welcome.
- Performance and cost constraints strongly suggest that we must use unverified and/or **untrusted components (e.g. COTS) in composite systems but still provide high assurance**. Research that incorporates formal verification and untrusted components that achieves a digitally assured result are also sought.
- Enable analysis and **identification of vulnerabilities in existing high-consequence systems** through application of model learning techniques for automated construction models corresponding to legacy codes.
- Application of machine learning techniques to enable **certified compiler optimization** for performance gains with correctness guarantees.

REFERENCES

[1] Samuel D. Pollard, Robert C. Armstrong, John Bender, Geoffrey C. Hulette, Raheel S. Mahmood, Karla Morris, Blake C. Rawlings, and Jon M. Aytac. 2022. Q: A Sound Verification Framework for Statecharts and Their Implementations. In Proceedings of the 8th ACM SIGPLAN International Workshop on Formal Techniques for Safety-Critical Systems (FTSCS 2022). Association for Computing Machinery, New York, NY, USA, 16–26. https://doi.org/10.1145/3563822.3568014