Highly Assured Computer Engineering

Shiu-Kai Chin
Department of Electrical Engineering and Computer Science
Syracuse University, Syracuse, New York 13244
chin@cat.syr.edu http://www.cat.syr.edu/~chin

Abstract

Engineering is fundamentally about the creation of structures with desired properties. Engineers strive to *calculate* properties of their implementations before actually building them. Our goal should be the elevation of professional practice in computer engineering to the point where system properties such as security are routinely calculated for structures of hardware and software. This goal requires: 1) the integration of design and analysis techniques using predicate calculus and mathematical logic into the engineering curriculum, and 2) devising a practical "linear-systems theory" for computer engineering. If successful, the consequences are: 1) *theorem-based* design would be as routine as simulation-based design, and 2) complexity could be managed by a theory of composition or "block diagram" algebra similar to the way linear systems theory and system transfer functions manage complexity in electrical engineering. There are historical parallels between the current need for assurance in system design and the challenge twenty years ago of making very large scale integrated (VLSI) circuit design commonplace. These parallels suggest some courses of action for achieving our goal.

1 Introduction

Engineering is about building things which work and knowing that what we build will work before actually building them. In the older engineering disciplines of civil, mechanical, and electrical engineering, the capacity to *compute* the properties of structures is well established. Statics is used to determine if a given structure of girders will support a given load. Laplace transforms are used to determine the stability of a structure of system-transfer functions. Algebra and calculus are the mathematical basis for engineering design and analysis.

Our increasing reliance on information systems of growing complexity move concerns of correctness, safety, reliability, and security to the forefront. By what means will we determine whether a system with many interconnected subsystems operating concurrently with varying degrees of synchronism or asynchronism is correct, safe, reliable, and secure *for all cases* of interest? Current design practice based on simulation is merely a form of virtual prototyping. Prototyping or "build-and-fix" methods alone are expensive and unreliable ways for determining the presence or absence of system properties.

If we were to go one hundred years into the future, would we see system design done using mathematical logic that is more sophisticated than propositional calculus? It seems likely we would. What would those design procedures look like? What training would engineers have who use the more sophisticated procedures? These questions must be answered in detail if we are to improve the level of practice in computer engineering.

The rest of this paper is organized as follows. Section 2 identifies composition and the need to refine theory into engineering practice as key challenges. Section 3 describes the parallel between the current situation and the area of VLSI design twenty years ago. We conclude in Section 4.

2 Challenges

2.1 Composition

The older engineering disciplines have well developed theories of composition. Civil and mechanical engineering use linear vector spaces. Electrical engineering uses linear systems theory based on the sinusoidal composition of signals. These notions of composition control complexity because structure can be abstracted away to behavior.

Until computer engineering has equivalent notions of composition which are routinely used, the timely design and verification of complex systems will be impractical.

2.2 Moving Theory to Practice

It is insufficient to say that mathematical logic should be used in system design without also having specific refinements to engineering practice. Modern hardware design is largely based on propositional calculus. Hardware designers are not taught symbolic logic in the same way logicians are taught logic. Designers are taught logic and state minimization techniques which are based on propositional logic. The emphasis is not on general theory. The emphasis is on logically well-founded design techniques relevant to design needs.

The use of predicate calculus as a design tool is completely lacking in engineering courses. No where is predicate calculus used to determine the behavior of designs in the same manner as the calculus of complex variables in linear systems theory is used to design communication and control systems.

It is insufficient to teach a single discrete mathematics course isolated from engineering design courses and expect engineers to make the leap from theory to practice by themselves. We cannot expect theorem-based design to displace simulation-based design when discrete mathematics texts give informal "proofs" instead of formal ones using the rules of predicate calculus.

If the designers of information systems cannot do logical calculations on their designs to test for the presence or absence of various properties such as security, how can we expect secure systems to be built reliably at all?

2.3 Critical Mass

There are typically at most one or two engineering faculty in each department who are competent in both engineering design and mathematical logic. They alone have insufficient mass to move formal methods and security into mainstream engineering at their institutions.

The only way to increase the number of faculty who use mathematical logic in engineering is to develop formal design methods as an extension of current design methods. There is a need for texts and courses which move beyond the simple exposition of mathematical logic. Texts and courses which apply mathematical logic within the context of design processes and design examples are needed. Examples of this in propositional logic include Kohavi's, *Switching and Finite Automata Theory*, as well as the countless texts on digital design.

Few engineers are interested in mathematical logic for its own sake. Virtually all engineers are interested in design. If practical design techniques using mathematical logic are developed, mathematical logic will be widely used. This will improve the level of practice in computer engineering.

3 Historical Parallels

Twenty years ago it was said, "there are fewer VLSI designers than there are NFL football players." Today, there are fewer designers trained in formal methods and security than there are NFL football players.

Today, there is no shortage of VLSI designers and VLSI design is part of almost every *undergraduate* computer engineering program. How did this happen and what can we learn from this?

VLSI design made a great leap forward because of the following:

- A confluence in the 1970s of semiconductor technology and the first generation of CAD tools for physical design.
- The book, *Introduction to VLSI Design*, by Carver Mead and Lynn Conway (of DARPA) made VLSI design simple and accessible to engineering faculty who were relatively unfamiliar with semiconductor processing but were competent in computer engineering.
- The Metal Oxide Semiconductor Implementation Service (MOSIS), sponsored by DARPA, taught faculty how to do VLSI design and promoted the teaching of VLSI design in U.S. universities by distributing CAD tools and fabricating student designs at little or no cost.

Today, much formal methods and security theory exists. There are many formal verification tools ranging from model checkers to theorem provers. There are security tools ranging from protocol-analysis using belief logics to theorem-provers with specialized theories for verifying non-interference properties of processes. These are the first generation CAD tools for high-assurance design. They are analogous to the first bullet of the above list.

What is missing are the other items in the list – texts which explain the use of mathematical logic as a design technology, and a recognized and impartial repository of tools and expertise whose mission is to promote and train the faculty who will train the next generation of engineers.

4 Conclusions

Theoretical research in mathematical logic and security along with their associated tools is still important. Nevertheless, the products of this research will have little or no impact on the level of professional practice unless the number of engineers who are capable of using such tools increases dramatically. This cannot be solved by creating tools which hide logical reasoning. This seems to be the operative meaning of, *"we will create verification tools which engineers can use,"* – a phrase which appears in many proposals for formal methods research. New tools alone will not solve this problem.

If we examine the engineering CAD tools which are widely used, we see that the tools provide support for a design method or technique which the engineer *understands*. Engineers can either accept, reject, or modify what these CAD tools do. There is no precedent in engineering where large numbers of engineers just use a tool without having a basic understanding of the underlying method the tool is supporting. It is unreasonable to expect the case would be different for formal methods, security, or assurance.

If we wish to improve the level of professional practice, we should develop:

1. Practical theories and techniques for composition, conceptually independent of any particular tool.

2. Usable design techniques based on mathematical logic, conceptually independent of any particular tool.

3. Texts and curricular support for formal design methods aimed at faculty whose area is design. Formal methods are a *means* to support system design rather than ends in themselves.

4. A MOSIS-like service for high-assurance design which supports training and infra-structure.

Should we worry about the engineers who are already in professional practice? No. Computer engineering is a field where the half-life of knowledge is from five to seven years. Those who adapt survive; those who don't are left behind. If high-assurance design becomes practical, a measure of its practicality will be how much of a competitive advantage it gives to those who know it. Newly graduating engineers proficient in new technologies typically have a lower cost-to-performance ratio than those engineers with obsolete skills who typically cost more and perform less. The costlier engineers with obsolete skills are replaced.

The engineering profession is not defined by what is done in commercial practice. It is defined by accredited engineering curricula. The level of professional practice will improve when we have practical high-assurance design methods which work – and when we train our students to use them.