

ABSTRACT

Modularity, that is the division of complex systems into less complex and more easily understood parts, is a pervasive concern in computer science, and hardware design is no exception. Existing hardware design languages such as Verilog and VHDL support modular design by enabling hardware designers to decompose designs into *structural* features that may be developed independently and connected together to form more complex devices. In the realm of high assurance for security, however, this sort of modularity is often of limited utility. Security properties are notoriously non-compositional, i.e. subsystems that independently satisfy some security property cannot necessarily be relied upon to maintain that property when operating in tandem.

The aim of this research is to establish *semantically modular* techniques for hardware design and implementation, in contrast to the conventional structural notion of modularity. A semantically modular design is constructed by adding “layers” of semantic features, such as state and reactivity, one at a time. From the high assurance aspect, semantic modularity enables different layers of semantic features to be reasoned about independently, greatly simplifying the structure of correctness proofs and improving their reusability. The major contribution of this work is a prototype compiler called ReWire which translates semantically modular hardware specifications to efficient implementations on FPGAs. In this dissertation I present the design and implementation of the ReWire compiler, along with a number of case studies illustrating both the practicality of the ReWire compiler and the elegance of the semantically modular approach to hardware verification.