Formal Modeling and Verification of Security Property in Handel C Program

Yujian Fu, Alabama A&M University, USA
Jeffery Kulick, University of Alabama in Huntsville, USA
Lok K. Yan, Air Force Research Laboratory, USA
Steven Drager, Air Force Research Laboratory, USA

ABSTRACT

Multi-million gate system-on-chip (SoC) designs easily fit into today’s Field Programmable Gate Arrays (FPGAs). As FPGAs become more common in safety-critical and mission-critical systems, researchers and designers require information flow guarantees for the FPGAs. Tools for designing a secure system of chips (SOCs) using FPGAs and new techniques to manage and analyze the security properties precisely are desirable. In this work we propose a formal approach to model, analyze and verify a typical set of security properties – noninterference – of Handel C programs using Petri Nets and model checking. This paper presents a method to model Handel C programs using Predicate Transition Nets, a type of Petri Net, and define security properties on the model, plus a verification approach where security properties are checked. First, a formal specification on the Handel C description using Petri Nets is extracted. Second, the dynamic noninterference properties with respect to the Handel C program statements are defined on the model. Third, in verification, a translation rule from the Petri Nets specification to the Maude programming language is also defined. Thus, the formal specification can be verified against the system properties using model checking. A case study of the pipeline multiplier is discussed to illustrate the concept and validate the approach.

1. INTRODUCTION

Multi-million gate system-on-chip (SoC) designs easily fit into today’s FPGAs. How to build a trustworthy computing system with high confidentiality of reconfigurable systems has been a broad research area for the past few years. Due to the ever increasing demand for higher speeds, lower memory and power requirements, it is increasingly difficult to transform a hardware description language (HDL) (Ghosh, 1999) into a bitstream format that can program the FPGA. This in turn increases the demand for the ability to verify security properties during the
design transformation. To define and verify the system properties precisely in a program, formal methods play a key role. However, there are few works on the formal specification of hardware description languages and formal verification of security properties. This work presents a systematic approach to the problem that includes a formal model of Handel C programs, security property specifications using linear temporal logic (LTL), and formal verification using a model checker. There are three steps. First, a Handel C program is formally specified using Predicate Transition Nets. Second, the security properties – noninterference – are defined on the model description. Finally, the model and properties are converted into the model checker programming language and the properties are verified against the system (Figure 1).

The main contributions of this paper are the modeling and specification of dynamic noninterference properties using Predicate Transition Nets (PrT Nets) and linear temporal logic (LTL) and formal verification of system properties on Handel C using a rewriting logic based model checker. This work establishes a platform for the formal verification and analysis of data flow security properties in Petri Nets using model checking.

The remainder of the paper is structured as follows. Section 2 presents background knowledge including FPGAs, Handel C, formal methods of Petri Nets and temporal logic, the dynamic semantics, security properties in FPGAs, and the principle of Maude, a model checker. Section 3 presents related works in formal verification using model checkers and discusses the limitations and benefits of existing verification tools. Section 4 introduces a PrT Net for Handel C programs based on an imperative Handel C syntax, defines the dynamic noninterference on the Handel C PrT Nets and the verification approach with meaningful results. Additionally a case study is used to validate the formal model and the results are discussed. Section 5 concludes the paper and presents future work.

2. PRELIMINARIES

2.1. FPGAs

The combination of flexibility and efficiency in the development of digital components has made the use of Field Programmable Gate Arrays (FPGAs) prevalent not only in space electronics (Katz, 2000) but also in more traditional avionic systems. An FPGA is a collection of programmable gates embedded in a flexible interconnect network that can contain several hard or soft microprocessors. FPGAs use truth tables or lookup tables (LUTs) to implement logic gates, flip-flops for timing and registers, switchable interconnects to route logic signals between different units, and I/O blocks for transferring data into and out of the device (Dubey, 2008). A circuit can be mapped to an FPGA by loading the LUTs and switch boxes with a configuration, a method that is analogous to the way a traditional circuit might be mapped to a set of AND and OR gates. An FPGA is programmed using a bitstream. This binary data, when loaded into the FPGA through specific I/O ports on the device, defines how the internal resources are used for performing logic operations.

Due to this flexibility and because FPGAs are standard components manufactured on a
Legal and Economic Justification for Software Protection
[www.igi-global.com/chapter/legal-economic-justification-software-protection/29538?camid=4v1a](www.igi-global.com/chapter/legal-economic-justification-software-protection/29538?camid=4v1a)

An Overview of RIAs Development Tools
[www.igi-global.com/chapter/an-overview-of-rias-development-tools/188204?camid=4v1a](www.igi-global.com/chapter/an-overview-of-rias-development-tools/188204?camid=4v1a)