Chapter XIV

Safecharts Model Checking for the Verification of Safety-Critical Systems

Pao-Ann Hsiung, National Chung Cheng University, Taiwan

Yen-Hung Lin, National Chung Cheng University, Taiwan

Yean-Ru Chen, National Chung Cheng University, Taiwan

Abstract

Unintentional design faults in safety-critical systems might result in injury or even death to human beings. However, the safety verification of such systems is getting very difficult because designs are becoming very complex. To cope with high design complexity, model-driven architecture (MDA) design is becoming a well-accepted trend. However, conventional methods of code testing and hazard analysis do not fit very well with MDA. To bridge this gap, we propose a safecharts model-based formal verification technique for safety-critical systems. The safety constraints in safecharts are mapped to semantic equivalents in timed automata. The theory for safety verification is proved and implemented in the SGM model checker. Prioritized and urgent transitions are implemented in SGM to model the safechart risk semantics. Finally, it is shown that
priority-based approach to mutual exclusion of resource usage in safecharts is unsafe and solutions are proposed. Application examples show the benefits of the proposed model-driven verification method.

Introduction

Safety-critical systems are systems whose failure most probably results in the tragic loss of human life or damage to human property. There are numerous examples of these mishaps. The accident at the Three Mile Island (TMI) nuclear power plant in Pennsylvania on March 28, 1979 is just one unfortunate example (Leveson, 1995). Moreover, as time goes on, there are more and more cars, airplanes, rapid transit systems, medical facilities, and consumer electronics, which are all safety-critical systems in our daily lives. When some of them malfunction or fault, a tragedy is inevitable. The natural question here is: Should we use these systems without a very high confidence in their safety? Obviously, the answer is no. That is why we need some methodology to exhaustively verify safety-critical systems.

Traditional verification methods such as simulation and testing can only prove the presence of faults and not their absence. Some methods such as fault-based testing and semiformal verification that integrates model checking and testing can prove the absence of prespecified faults. Simulation and testing (Sommerville, 2004) are both required before a system is deployed to the field. While simulation is performed on an abstract model of a system, testing is performed on the actual product. In the case of hardware circuits, simulation is performed on the design of the circuit, whereas testing is performed on the fabricated circuit itself. In both cases, these methods typically inject signals at certain points in the system and observe the resulting signals at other points. For software, simulation and testing usually involve providing certain inputs and observing the corresponding outputs. These methods can be a cost-efficient way to find many errors. However, checking all of the possible interactions and potential pitfalls using simulation and testing techniques is rarely possible. Conventionally, safety-critical systems are validated through standards conformance and code testing. Using such verification methods for safety-critical systems cannot provide the desired 100% confidence on system correctness.

In contrast to the traditional verification methods, formal verification is exhaustive. Further, unlike simulation, formal verification does not require any test benches or stimuli for triggering a system. More precisely, formal verification is a mathematical way of proving a system satisfies a set of properties. Formal verification methods such model checking (Clarke & Emerson, 1981; Clarke, Grumberg, & Peled, 1999; Queille & Sifakis, 1982) are being taken seriously in the recent few years by several large hardware and software design companies such as Intel, IBM, Motorola, and Microsoft, which goes to show the importance and practicality of such methods for real-time embedded systems and SoC designs. For the above reasons, we will thus employ a widely popular formal verification method called model checking for the verification of safety-critical systems that are formally modeled.