



REAL-TIME SYSTEMS—Zhen Zhu computer science Ph.D. student, discusses research with Dr. Albert Cheng, director of the Real-Time Systems Laboratory, in methodologies for verifying the timing properties of space vehicles. Their work will affect the safety of astronauts in the Crew Return Vehicle (CRV) of the International Space Station (ISS).

Timing Analysis and Scheduling of the X-38 Space Station Crew Return Vehicle and Other Space Vehicles

Albert M. K. Cheng, Ph.D., Director, Real-Time Systems Laboratory; Associate Editor, *IEEE Transactions on Software Engineering*; Department of Computer Science; Senior Member, IEEE

<http://www.cs.uh.edu/~acheng/acheng.html>

78-ISSO

Abstract—

This project investigates the analysis and verification of the timing properties of space vehicles. This is a departure from the usual practice of software and system testing, which can only uncover errors but cannot guarantee the absence of errors. UH researchers have analyzed the timing properties of the avionics of the X-38 autonomous spacecraft currently being designed and built by NASA as a prototype of the International Space Station (ISS) Crew Return Vehicle (CRV).

The CRV will be permanently attached to the ISS with the capability to automatically and safely bring to earth a crew of seven passengers in the event of an emergency ISS evacuation. To verify the planned performance of the safety-critical system functions, an initially high-level, and, later, more detailed specification of the X-38 multiprocessor system task structure can be modeled in Real-Time Logic (RTL) and Presburger Arithmetic representations.

This successful application of formal verification strategies to the X-38 leads to the next step of developing a comprehensive framework and toolset for the analysis and verification of space vehicles and airplanes.

REAL-TIME SYSTEMS OFTEN OPERATE IN AN ENVIRONMENT with stringent safety and response time constraints. These systems include airplane avionics, medical monitoring instruments, smart robots, space vehicles, and other safety critical applications. In addition to functional correctness, these systems must also satisfy stringent timing constraints. The result of missing a deadline in these systems may be catastrophic. If a given real-time system cannot deliver an adequate performance in bounded time, it must be optimized or resynthesized. Many formal analysis and verification techniques for real-time systems have been developed in the past two decades, but they are seldom practically applied to a large-scale industrial system. Furthermore, commercial scheduling tools based on sound real-time scheduling theory are also becoming more available and need to be evaluated by applying them to the task of scheduling a large-scale system.

There are two ways of ensuring system safety and reliability. One way is to employ engineering (both software and hardware) techniques such as structured programming principles to minimize implementation errors and then utilize testing techniques to uncover errors in the implementation. The other way is to use formal analysis and verification techniques to ensure that the implemented system satisfies the required safety constraints under all conditions, given a set of assumptions. In a real-time system, not only do we need to satisfy stringent timing requirements, but we must be able to guard against an imperfect execution environment which may violate pre-runtime design assump-

tions. The first approach can only increase the confidence level we have on the correctness of the system because testing cannot guarantee that the system is error-free. The second approach can guarantee that a verified system always satisfies the checked safety properties.

Technical Plan and Equipment

Testing is perhaps the oldest technique for detecting errors or problems in implemented software, hardware, or non-computer systems. It consists of executing or operating (in the case of a non-computer system) the system to be tested using a finite set of inputs and then checking to see if the corresponding outputs or behavior are correct with respect to the specifications. To test a real-time system, the values as well as the timing of the inputs are important. Similarly, both the output values and the time at which they are produced must be checked for correctness.

Testing is good for revealing errors in the tested system but usually cannot guarantee that the system satisfy a set of requirements. To apply formal verification techniques to a real-time system, we must first specify the system requirements and then the system to be implemented using an unambiguous specification language.

Once these specifications are written, the formal methods expert can verify whether the system specification satisfies the specified requirements using his/her favorite formal verification methods and tools. These formal methods and tools can show the satisfaction of all requirements or the failure to satisfy certain requirements.¹

They may also pinpoint areas for further improvement in terms of efficiency. These results are communicated to the applications expert who can then revise the system specification or even the system requirements. Formal specifications are next revised to reflect these changes and can be analyzed by the formal methods expert again. These steps are repeated until both experts are happy with the fact that the specified system satisfy the specified requirements.

Experimental Activity

Researchers focus on the timing within the quad-redundant embedded avionic control units, called Flight Critical Computers (FCC) 1 through 4, of the preliminary X-38 Vehicle 201 (Fig. 1) data system architecture. These units receive all sensor input values, provide embedded guidance and application processing, and control actuation. Each FCC is a unit comprised of two PowerPC (PPC) processors, Input/Output cards, and several other devices in a VME chassis. The first processor, called the Flight Critical Processor (FCP), houses all application software such as guidance, navigation, and control. The second processor, the Instrumentation Control Processor (ICP), is responsible for assembling inputs from all other sensors and sending the data over the VME backplane to the FCP for processing. Both processors run the VxWorks real-time operating system as well as specially-developed system services.

Timing Properties

For purposes of safety and verifiability, all X-38 avionics design efforts have focused upon designing a system that is highly deterministic. The quad-redundant design of the four FCC's relies on Byzantine agreement (a voting and message-passing protocol) to tolerate a single Byzantine fault. Because of this design, tasks are required to run at the same time in all processors, with results of

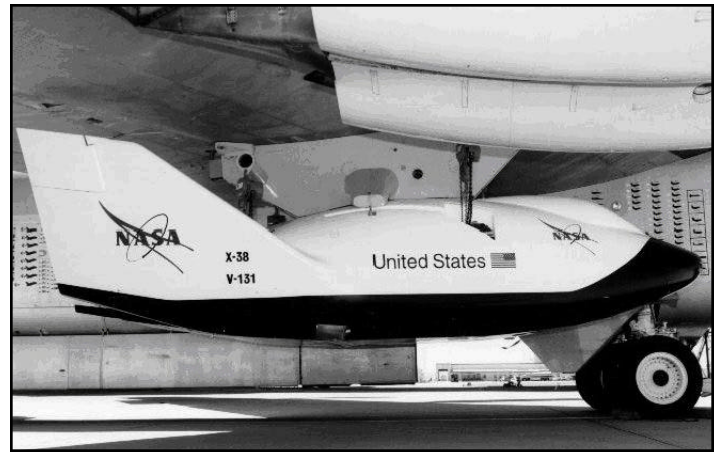


Figure 1. X-38 vehicle 131 captive-carry.

their processing being voted every 20 ms “minor” frame. The ICP and FCP processors are thus synchronized to the same 20 ms processing frame. Another example of similar real-time fault-tolerant avionics is the quad-redundant fly-by-wire flight control of the Lockheed F-117A stealth fighter aircraft. Other examples of fault-tolerant avionics include the Boeing 777 Integrated Airplane Information Management System, and the Airbus 340 Flight Warning System.

We consider a snapshot of the anticipated task timing relationships for the X-38 vehicle. The most critical control loop begins 18 ms into the processing frame where the ICP inputs all 50 Hz (cycles per second) sensor data. These data are passed to the FCP which reads the sensor data, processes it, and provides output actuator commands back to the ICP. The ICP then reads and issues those commands to affected actuators. In order to effect safe guidance of the vehicle, this whole processing loop is required to complete within 10 ms. To ensure this type of deterministic processing, tasks are generally assigned fixed priorities based upon the Rate-Monotonic Scheduling (RMS) algorithm.² Scheduling is partially accomplished with the aid of a cyclic executive.

Discussion

Timing and Safety Analysis using RTL analyze and verify system timing properties. UH researchers use RTL described in this chapter to specify the safety-critical aspects of the X-38 system. As we have seen, RTL allows timing and safety analysis which may flexibly be used for “what-if” scenarios as well as life-cycle system verification, and may be extended to represent broader aspects of the X-38 avionics system. The two X-38 flight-critical control loops, as well as one non-flight critical control loop and the associated safety assertion, have been modeled. After converting the RTL representation into a Presburger Arithmetic format and ultimately a constraint graph, cycle analysis verifies safety assertion satisfaction.

Results: Constraint Graph Analysis

In order to verify the satisfaction of the safety assertion, Presburger formulas are represented in a constraint graph shown in Fig. 2. The system specification alone produces a graph with no positive cycles. Negation of the safety assertion, however, yields edges which produce positive cycles between clusters, thus verifying critical system performance. For example, a

