Introduction
Designing effective error handling in an embedded software systems is essential for acceptable and reliable functionality in cases of errors and for the recovery from faults.

Errors in the error handling system can cause catastrophic failures of the software and can endanger human life. We take a principled approach of extending a model of computation (MOC) with timing semantics for embedded systems by an error handling mechanism for timing errors in model-based design.

Background
It is extremely important that one identifies and is capable of handling error cases before deploying embedded software.

One ineffective solution that has been used in the past is to have the system reset itself for every error encountered. If a system resets itself too often this can lead to significant loss of productivity, possible loss of data, and in extreme cases, possible loss of life.

Examples of the importance of error handling:

In 1996 the European Space Agency’s Ariane 5 rocket self-destructed 40 seconds after launch. The underlying cause of the self-destruct sequence was a 64-bit floating point to 16-bit integer conversion exception. This occurred because of reuse of code designed for the much smaller Ariane IV [1].

More recently, the SPIRIT Mars rover encountered a “reboot loop” shortly after landing, where a fault during the booting process caused the system to reboot again. Luckily, a software patch solved the problem and the mission continued successfully[2].

A Royal Airforce pilot accidentally dropped a practice bomb on the flight deck of the Royal Navy’s aircraft carrier. It missed its intended target and several sailors were injured. The cause was attributed to a timing delay in the software.

In an effort to avoid possible similar mistakes with newly designed systems and to provide a more systematic means of dealing with timing errors, we present preliminary work that extends a model of computation (MOC) for embedded systems which features timing semantics.

Methodology

The focus of this work is on timing errors. A timing error occurs when the specification says one thing and the implementation does something else. Often times, this is caused by execution at a time that violates a specification.

Model-based design, simulation, and synthesis is being used more than before in lieu of hand writing code and testing it [3].

There is also a resurgence in Cyber Physical System design due to renewed interest in the area. If these trends continue, we will see:

1. More use of model-based design with timing specifications in the design of Cyber Physical Systems;
2. The desire to include error handling explicitly in a model instead of in an ad hoc manner.

Objective

Add meaning to what is done in the event of a timing error. We achieve this by:

1) Extend concepts from real-time programming languages to model-based design.
   * Exception handling
2) Adding concepts to hierarchical state machines
   * Error Transitions

Methodology

Adding in other types of timing error

As

More recently, the SPIRIT Mars rover encountered a “reboot loop” shortly after landing, where a fault during the booting process caused the system to reboot again. Luckily, a software patch solved the problem and the mission continued successfully [2].

A Royal Airforce pilot accidentally dropped a practice bomb on the flight deck of the Royal Navy’s aircraft carrier. It missed its intended target and several sailors were injured. The cause was attributed to a timing delay in the software.

In an effort to avoid possible similar mistakes with newly designed systems and to provide a more systematic means of dealing with timing errors, we present preliminary work that extends a model of computation (MOC) for embedded systems which features timing semantics.

Future Work

2) Incorporating representative probabilistic distributions into the timing
3) expanding the preliminary work in C and Java code generation

There is also a resurgence in Cyber Physical System design due to renewed interest in the area. If these trends continue, we will see:

1. More use of model-based design with timing specifications in the design of Cyber Physical Systems;
2. The desire to include error handling explicitly in a model instead of in an ad hoc manner.

References


Acknowledgements

This work was supported in part by the Center for Hybrid and Embedded Software Systems (CHESS) at UC Berkeley, which receives support from the National Science Foundation (NSF awards #0720882 (CNS ENS:PRET) and #0720841 (CNS-CP)), the U. S. Army Research Office (AROW9311NP-07-2-0019), the U. S. Air Force Office of Scientific Research (MURI #FA9550-06-1-0212), the Air Force Research Lab (AFRL), the State of California Micro Program, and the following companies: Agilent, Bosch, HSBC, Lockheed Martin, National Instruments, and Toyota.

This work was also supported by the Jenkins Pre-doctoral Fellowship program as well as a Jenkins Pre-doctoral Fellowship Program Mini Grant