One millisecond too late is too late!

For how long can a computer program *really* run? Such information is necessary when constructing computer systems to control potentially dangerous products like cars, airplanes, and nuclear power plants. At Uppsala University, tools are being developed to support the generation of timing guarantees, thus helping industry build safer products faster.

Mercedes A-klasse and JAS 39 Gripen are examples of products depending on advanced time-critical computer systems to function correctly

Over the last few decades, our society has become increasingly dependent on computers. This not only includes the gray PC boxes at our desks, but also the myriad of computer systems being embedded into everyday things around us. In fact, 99 percent of all computers sold are used to control vehicles, appliances, power plants, telecommunications equipment, toys, and other products that are intrinsic parts of modern society.

Take a look around in a modern car. There is an embedded computer controlling the engine, keeping performance up and fuel consumption down by very precisely controlling the ignition and fuel pump. For your safety, anti-lock brakes (ABS) are installed, controlled by embedded computers that continuously monitor the behavior of the car and avoid brake locking. In the unlikely event of a collision, yet other embedded computers will detect the crash within milliseconds and deploy the airbags.

Timing is critical

Such embedded computer systems are built from one or more computers. On each computer, one or more computer programs are running, all of which might communicate with each other to fulfill the system functionality. Failures of these embedded computer systems could endanger human life and substantial economic values, and thus there is a need for software development methods and tools that minimize the risk of failure. Timing is a critical parameter that has to be considered: detecting a collision a millisecond too late is too late, and we want to know that this will never happen.

Researchers have investigated how to derive timing information for computer programs, and how to use this information to provide guarantees of system behavior, thus providing a solid foundation for constructing safer products. Since a system must always work even in the most stressful situations, information regarding the worst case is usually needed, and deriving such information is not trivial.

Using formal mathematical models, automatic analysis tools can determine the worst-case execution time (WCET) of a program without actually running it. Such methods rely on proven models of program behavior and timing to generate safe WCET estimates that never underestimate the actual worst-case. It is a concept similar to inspecting the blueprints of a bridge to determine whether it will fall down or not, instead of building it and driving heavy trucks across it.

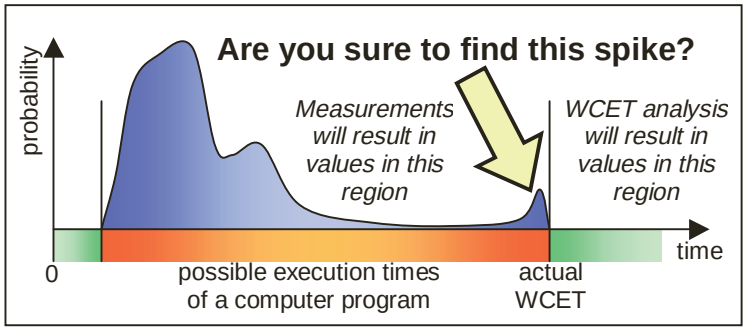
Hardware and software

The problem that is solved by WCET analysis is that a computer program typically does not have a single fixed execution time, due both to the characteristics of the work it has to perform and the hardware on which it runs.

Computer programs tend to be sensitive to their inputs. Think about sorting a deck of cards: if all cards are already sorted, this is very quick. However, if they are well shuffled, it can take quite some time. The number of cards to sort is also important: sorting two cards is faster than sorting the whole deck. Thus, the same software (computer program) can take different amounts of time depending on the situation.

The hardware the program runs on is just as important. Obviously, a program runs much faster on a brand new PC than an old computer, and this has to be taken into account. For embedded systems, there are hundreds of different varieties of computers to choose from, each specialized for a certain type of task. A computer that is great at processing radio signals in mobile phone systems can be really bad at detecting car crashes, and these differences have to be taken into account.

Better than testing



The traditional way to determine the timing of a program is to run it many times and try different input values to try to find the worst-case execution time. This is time-consuming and boring work, which still does not guarantee the quality of the results. Imagine trying to find the particular ordering of cards that will take the longest time to sort for a particular sorting method. It is very hard to know when you have found the worst case without trying all orderings. If any change occurs to the hardware or software, all testing work has to be redone.

WCET analysis avoids the need to run the program by simultaneously considering the effects of all possible inputs, and how the program interacts with the hardware. This is done by using mathematical models of the software and hardware involved. Sometimes, it is necessary to make pessimistic assumptions (if we do not know how many cards we need to sort, we have to assume a full deck). The result is a worst-case execution time estimate that is greater than or equal to the actual worst-case that can occur, and thus safe in all circumstances. The analysis must be redone after a change to the hardware or software, but the amount of work involved is much smaller than for testing. Thus, WCET analysis can help companies put safer products on the market quicker.

Research in Uppsala

The execution time analysis project in Uppsala has focused on design a very general WCET analysis tool that can be applied to a wide spectrum of different embedded computers and programs. The results so far are encouraging and internationally competitive, and the long-term goal is to create a tool that can be used in industrial settings. Case studies in industry have indicated the real-life usefulness of WCET analysis tools.

For more information, contact: Jakob Engblom (jakob.engblom@docs.uu.se, 018-471 73 44) or Andreas Ermedahl (andreas.ermedahl@docs.uu.se, 018-471 31 72),
Department of Information Technology, Uppsala University, Box 337, 751 05 Uppsala, Sweden.