User Story

CompCert, StackAnalyzer, and aiT Take Off

TU Munich, Institute of Flight System Dynamics

Diamond DA42 NG Airplane

The Institute of Flight System Dynamics at the Technical University of Munich operates a Diamond DA42 NG as a flying testbed. Its purpose is to test the institute's research in the area of flight control and navigation algorithms in a real environment. The aircraft is modified and equipped with an electro-mechanical fly-by-wire flight control system. To ensure safe operation under all conditions, the safety pilot on board can at any time decouple the fly-by-wire system from the original mechanical flight control system.

This modification enables the Institute to perform flight tests of newly developed autopilot and flight management functions — or even carry out fully automated flights — without passing a complete certification of the system. Nevertheless, the processes applied for the development of flight control laws and software of the flight control computer (FCC) are oriented towards the applicable rules and standards for certification, like the ARP 4754A or the DO-178C.

The Institute developed the flight control laws of the FCC using a model-based approach with Matlab Simulink/Stateflow. The developed algorithms are automatically translated into C source code using Embedded Coder. The code is then embedded into a framework that provides a runtime environment, handling the interface data and issuing a cyclic call to the execution of the control laws.

Right now the FCC software has some 125k lines of code, of which 92% are auto-generated.

During the development of the control laws, assumptions about the maximum delay of the control loop are made. This entails hard real-time requirements for the FCC.

Verified Compilation with CompCert

At first, GCC was used to generate the executable object code for the FCC without any optimizations. However, as more functionality was added, it became difficult to stay within safe execution-time bounds with GCC -O0.

Additionally, the need to comply with MathWorks' DO Qualification Workflow and the wish to run automatic verification tools like Simulink Code Inspector restricted the optimization capabilities of Embedded Coder.

This made the Institute consider a move to the optimizing C compiler CompCert.

For the migration, no changes had to be made to the auto-generated source code. In the manually developed framework, changes had to be made to parts that used bit fields or relied on the exact memory mapping of structures:

✐ For certain combinations of data types, the memory layout of structures differs between GCC and CompCert. Therefore, all parts requiring a certain memory address to be read or written have been rewritten to avoid structures.
✐ Bit fields are not supported by CompCert per default. It is possible to enable them, but then the compilation is no longer formally proven correct. To ensure verified compilation, all bit fields have been replaced. This has the added benefit that the code is now compliant with MISRA C.

Currently, the executable object code generated using CompCert is under test in a hardware-in-the-loop simulation. In comparison to the executable object code of GCC without optimization, the execution time was reduced by approximately 50%.

✐Stack Usage Analysis with StackAnalyzer

StackAnalyzer is used in order to ensure that the memory consumption by the stacks stays within the given bounds.

As no full software testing takes place, it is impossible to derive the memory bounds for the stack through measurements alone.

Thus, StackAnalyzer provides a quick way of proving that the space reserved for the stack is sufficient and no unexpected behavior will occur due to an overlap of the stack with other memory sections.

Timing Verification with aiT

To verify that the timing requirements allocated to the FCC are fulfilled, the software is analyzed using aiT WCET Analyzer, which computes a safe upper bound for the execution time.

The calculation of the flight control laws is carried out on an MPC 8349. This processor integrates an e300 core with data and instruction caches and a four-stage pipeline. In order to obtain the desired performance, both the instruction and the data cache are enabled. Therefore, the execution time of an instruction depends on the history of instructions executed before it. This makes it impossible to determine the WCET by measurements only.

As the analysis is performed on the fully linked executable, the tool cannot always automatically determine the bounds of loops. In such cases, the bounds have to be annotated by the user. However, as far as the auto-generated code is concerned, very few annotations are actually necessary. At the moment, only the functions generated for the calculation of values interpolated from lookup tables need to be annotated. These functions are part of the so-called shared utilities of Embedded Coder. They have canonical names and are always the same if a certain type of a lookup table is contained in the model from which the code is generated. The bound of the loop searching through the indexes of the table is dependent on the function arguments. With an AIS2 annotation, it is possible to define the loop bound depending on these arguments. Thus the annotation can be kept universal and does not need to be adapted if the parameters or the width of a lookup table are changed.

Embedded Coder does not provide its own implementation of the math functions like square root or sinus, and therefore calls the library functions of the applicable system. Currently Newlib is used for the calculation of math functions, a free C library intended for use on embedded systems. Most math functions consist of a range reduction algorithm and an algorithm calculating the function value for a limited array. The loop bounds of the range reduction algorithms sometimes cannot be automatically determined and need to be annotated. Since functions in Newlib do not fulfill the requirements for software that shall be certified, the libraries are currently analyzed and will be replaced by a custom implementation fulfilling certain needs in the future. Then, the WCET of the math functions will have to be considered as well. Should the new functions still contain loops that cannot be automatically analyzed, the Institute will make use of aiT's support for annotations provided as comments directly in the source code. This way, the information will have to be maintained in just one place.

The Verdict

"With CompCert it is possible to decrease the execution time of the flight control algorithm by a significant amount. The reduction of the execution time can be used for additional functionality. With the WCET analysis of aiT and the StackAnalyzer it is possible to do a fast verification of the software. For the given CPU architecture a static analysis is the only practical way to determine a safe upper bound for the execution time, while the stack analysis ensures that the space reserved for the stack pointer is sufficient without a complete software test."

K. Nürnberger, TU Munich, Institute of Flight System Dynamics