Software Performance Optimization Abstract

Dr. Max W. Lee and Marshall L. Moluf

Reducing software execution time for embedded systems is always challenging because enhancements must not impact system functionality. Optimization efforts usually occur during integration when the operational hardware is unavailable or is being used to verify another software activity. Considerations include software processes employed, achievable gains, technical risk of parallel baselines, maintaining design structures to facilitate testing, migrating optimized code into baselines, and impacts to schedule/cost. Multiple software engineering practices may be used to mitigate these concerns.

Traditional enhancements include combining loops, employing more efficient conditionals, unrolling short loops, in-lining procedures, and utilizing global literals. Occasionally, optimizing as few as five lines of code results in significant performance improvements, especially in high usage procedures. However, sometimes code changes expected to result in faster execution times are instead slower when using the target processor. Additionally, some optimizations that seem to run slower in a development environment, such as Windows NT 4.0, actually run faster on the target processor. Decisions on whether to include or exclude these changes are made on a case-by-case basis after repeated timing measurements on the test bed processor.

Modifying code is only one aspect of the optimization process. Selecting what to modify, analyzing impacts of coding changes on the system, ensuring accuracy of changes, and planning how to integrate the changes are equally important! This presentation describes tools for identifying viable enhancement opportunities, developing the optimization process, exercising operational states, baselining benchmarks, simulating/emulating hardware, verifying optimized functions, measuring timing limitations when using a host computer instead of the target processor, and the importance of using a basic test bed.

The techniques described above were used in an actual optimization effort. The scope of the effort spanned over 35,000 lines of previously integrated C code. Seven out of 47 modules were optimized with no changes made to functionality of the optimized software. Key lessons learned are shared along with best practices identified during and after the effort.

Dr. Lee has been with Raytheon since 1983 where he has managed signal processing programs and manufacturing organizations. Prior to joining Raytheon he managed electronic warfare requirements and software development groups, and implemented the software for inertial navigation and avionics systems. Mr. Moluf has been with Raytheon since his graduation from Kansas State in 1999, developing software for several embedded systems. Both authors are currently optimizing real-time software for an embedded processor.