Smaller, faster, better

“The profile can be used to answer questions about system behaviour such as ‘where is my application spending most of its time?’.”

How development tools can help software engineers to produce more efficient code for embedded systems.

By **Ken Butler**.

As more functionality is moved to single processors, embedded software developers are having to build in more functionality more quickly and with greater performance. In this kind of environment, they need tools that provide easier and more robust methods for developing and optimising systems.

No longer can a developer get along with a basic assembler, simple linker and rudimentary debugger. Embedded software development today requires not only support for high level languages, such as C/C++, but also ways to see how software and target processor interact in order to help debug and optimise programs.

One modern architecture that can take over many functions in a system is Analog Devices' Blackfin processor. Blackfin combines signal processing attributes, like dual MACs, with classic risc processor characteristics, such as a memory management capability that facilitates simplified microprocessor type programming modes and styles.

To support program development for Blackfin, CROSSCORE – Analog Devices' development tools product line – provides the VisualDSP++ integrated development and debug environment (IDDE) and EZ-KIT Lite evaluation systems, as well as pci and usb emulators for rapid on chip debugging.

Statistical profiling is one feature that can speed the development and optimisation of programs. Traditional profiling techniques rely on instrumenting user code in order to gather statistics on program execution performance. The instrumentation code can affect system performance, especially in highly tuned code, and often resources aren't available for capturing the data. With statistical profiling, the debugger can sample the program counter non intrusively while the processor is running. The longer the sampling time, the more accurately the statistical profile will match the actual profile.

The profile can be used to answer questions about system behaviour such as 'where is my application spending most of its time?' and 'how much time is spent executing any particular function, high level language instruction, or assembly level instruction?'.

The EZ-KIT Lite system comes with an evaluation suite of the VisualDSP++ debugger. This has a window that can be used to display the profile with varying levels of granularity. Each program function is shown, along with the percentage of total run time consumed by that function. The C source for the function is also shown and the assembly instructions associated with each line of C code and the contribution for each assembly instruction are available.

With this information, a programmer can quickly identify what parts of the program make the largest contributions to overall performance. Developers can then concentrate their efforts accordingly.

But speed isn't the only design criterion; program size can be just as important. The C/C++ compiler includes options to control optimisations to favour size or speed, as well as pragmas (#pragma optimise_for_speed, #pragma optimise_for_size) that control optimisation decisions on a function basis.

For example, a developer may want to generally optimise for space, while optimising for speed a core processing function that the profiler highlighted as a significant contributor to execution time.

Feedback about program size is provided to users by the linker in a map file. VisualDSP++ also presents this information graphically through the memory interface in the Expert Linker. The post link display shows each section that was mapped into memory, as well as its size. When profiling data is available, the Expert Linker will use colour to identify which sections are program 'hot spots' – the redder, the hotter.

In an embedded system with a memory hierarchy, where access to some memory locations is slower than access to others, this display can highlight code that executed frequently from external memory. Using the same Expert Linker display, the developer can drag and drop a code section from one memory segment to another.

The full VisualDSP++ toolset includes simulators for all processors. There are super fast functional simulators that can develop and test algorithms before a hardware platform is available. And there are cycle accurate simulators that model all behaviour of the core processor.

The cycle accurate simulator can be used to get exact (as opposed to statistical) profiling information. It can also be used to understand the behaviour of the hardware below the instruction level.

To reach high clock rates, processors execute instructions in a pipeline that operates much like an assembly line in a factory. At each clock cycle, instructions move from one stage to the next. As long as the pipeline stays full, the processor will effectively execute an instruction each cycle. If something happens to stall an instruction in the pipe, such as needing data from a previous instruction, effective throughput will be lower.

To achieve maximum performance, programmers need to see and understand the pipeline. The VisualDSP++ debugger has a Pipeline Viewer that follows the migration of each instruction through the pipeline, highlighting 'bubbles' that can affect performance.

In the Pipeline Viewer, change of flow is also monitored because jumps and branches can invalidate instructions that are in the pipeline but are not on the path that is taken. Stalls, bubbles and kills are all emphasised, with detailed information available with mouse over, so the developer can get an understanding of how the program interacts with the processor.

The C/C++ compiler has many ways for a user to control low level system behaviour whilst programming with a high level language. It has compilation controls at invocation to direct the compiler to optimise and whether to favour size or speed. In addition to the pragmas mentioned above, other pragmas give the compiler information that can be used to generate more efficient code. There are also compiler built in operations that allow the user to read and write system registers. And there is the ability to insert assembly instructions directly into the program if needed.

The compiler can generate its best code when it has the most information about how a program will behave. The VisualDSP++ C/C++ compiler has a Profile Guided Optimisation feature that gets direct feedback on the running of a program to make decisions on how best to optimise.

The IDDE guides users through the steps needed to use Profile Guided Optimisation effectively. The user first builds a version of the program that is instrumented to collect information the compiler can use for optimisation decisions – for example, which sides of branches are taken most frequently or whether most vectors processed are even length.

The program is then executed in the simulator using a set of 'training data'; the more training data matches actual data, the better the optimisations will be. Profile data is then fed into the compiler when the program is rebuilt and the compiler will generate code that executes fastest for the representative data.

Compiler improvements can be dramatic for control code, where there may be tests for error conditions which should never occur under normal circumstances. The compiler can retain the code for the error cases whilst optimising the error free path through the program. With signal processing code, the compiler can choose loop optimisation techniques best suited for the data the program will process.

As embedded processors handle more system functionality, better and more effective tools are needed to help users quickly develop and optimise performance. VisualDSP++ is one example of a tool suite that gives the user many ways to visualise performance. And with Profile Guided Optimisation, the compiler can make direct use of system performance information to optimise programs

Analog Devices' CROSSWARE tool range includes high performance usb based emulators to provide a portable, non intrusive target based debugging solution.

Author profile:

Ken Butler is Assembler and Simulator Development Manager for Analog Devices.

Take a free 90 day test drive of VisualDSP++ by downloading a test drive or requesting a cd at: www.analog.com/processors/tools/testdrive

ANALOG DEVICES

Sponsored Tutorial

For more information on Analog Devices' CROSSCORE development tools, go to www.analog.com/processors/tools