

Embedded Technology

Debugging with Hardware Trace Data

As the complexity of embedded systems continues to increase, there is a need for better debugging tools to speed development time and lower development cost. One technology that is becoming more and more popular to help address these issues is hardware trace data, which can be used to easily track down some of the hardest bugs in embedded development today. Whereas trace has traditionally been used primarily as a tool to track down hardware-level problems, it is now ready to move beyond these specialty applications and become a standard software debugging tool. This article discusses some of the primary benefits of trace data for debugging problems encountered by software engineers developing embedded applications. These benefits include additional visibility into the execution of a program, as well as assistance in tracking down all forms of bugs in embedded software.

Hardware trace consists of essentially a history of the instructions executed by a microprocessor. It may also include context-switches between various tasks and even addresses and values written to or read from memory. This information is generally streamed out of a dedicated trace port on a microprocessor and is collected by a trace collection device, such as the Agilent E5904B Trace Port Analyzer (www.agilent.com) or the Green Hills SuperTrace Probe (www.ghs.com). Because dedicated logic is used to output trace data, the collection of this information has no impact on the actual execution of the processor. This enables trace data to be collected in a completely non-intrusive fashion.

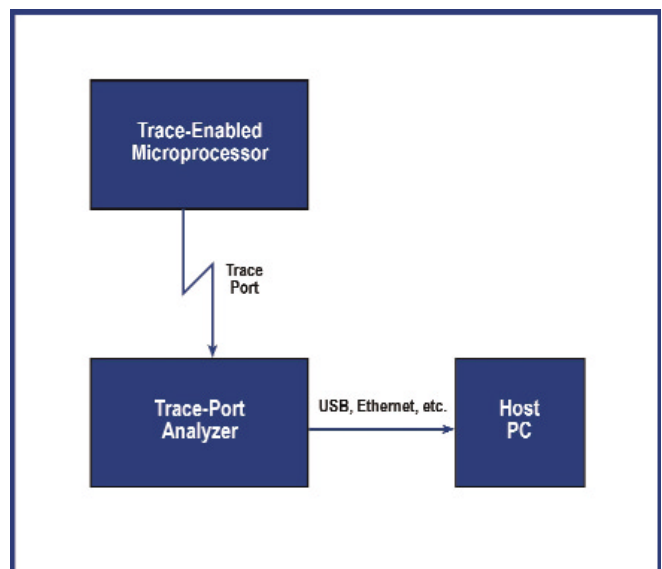
After it has been collected, the trace data is presented to a software engineer for analysis to help track down problems with their embedded software. Advanced trace analysis software is available that enables software engineers to use a familiar software debugger and other high-level tools to view the execution of their programs. For instance, tools such as the TimeMachine debugger from Green Hills Software enable users to collect trace

information and then step forwards and backwards through the actual execution of their program. The analysis software can infer additional information about the state of the system, including register and memory values, allowing nearly complete visibility into the system while the processor ran at full speed. Figure 1 shows a block diagram of the trace collection process.

A growing number of embedded processors support hardware trace. Some CPUs that support trace include the ARM family of processors, the PowerPC 4xx family from IBM, as well as the PPC5554 and MAC7111 CPUs available



by Michael Lindahl,
Green Hills Software



This figure shows the components of a debugging session using a trace-enabled microprocessor. The trace data streams off of the CPU and is collected by the Trace Port analyzer. This information is then passed on to a host PC for analysis by a software engineer.

from Freescale. Each of these chips provides a different trace port with slightly different features, but each trace port can be used to easily debug complex software bugs. These chips offer debugging environments that make solving many of the most difficult software problems significantly easier.

One class of difficult bugs that often plagues embedded software development is race conditions. Since these bugs often occur when subtle timing requirements are violated, they may not be easily reproducible and may appear as random glitches in the system. Traditional debugging techniques often alter the real-time characteristics of a system. For instance, instrumenting code with print statements to capture an execution log often alters the timing of the system enough that it can cause a bug to disappear while attempting to track it down. However, with hardware trace, if you can capture a single instance of a bug in your program, then you can use the analysis tools to track down the root causes and eliminate them, even without creating a reliable and reproducible test case. Since capturing a trace buffer is completely non-intrusive, debugging with trace allows you to track down and fix difficult bugs without worrying about the effects that debugging may have on your system.

Another major benefit that hardware trace provides is non-intrusive performance analysis. This allows system designers to track down which components of their software are using too much CPU time. Additionally, trace allows very accurate measurement of interrupt latencies and context switch times as well as other common system bottlenecks. This information is very useful for optimizing system performance, whether to lower the cost of the processor required, to minimize power consumption or to squeeze extra features into an existing design. Whereas traditional performance analysis tools require code instrumentation or other measurement techniques that modify the run-time charac-

teristics of a system, trace allows you to benchmark and investigate the performance characteristics of your production software without rebuilding or modifying your program. This also means that you can perform coverage analysis or other system verification, again without the need to modify the real-time performance of your system.

While hardware trace is useful for solving a wide variety of software problems, it is particularly well-suited to the development of embedded systems. This is because designing embedded systems often involves meeting critical timing requirements. Additionally, since the processors used on these systems are often selected based on their ability to meet specific requirements, there is often insufficient processing power left over for instrumentation and debugging. Using a dedicated trace port allows software engineers to easily gain insight into the execution of their software without suffering any overhead.

As trace analysis tools continue to improve, more information will be available to software engineers to help them solve the most difficult software bugs as easily as possible. Hardware manufacturers have also shown great interest in expanding the number of microprocessors that support trace debugging. As more and more chips implement dedicated trace ports, this information will be available to more and more engineers, enabling more efficient and effective software development. Hardware trace is already enabling many software engineers to achieve faster time to market with lower costs, and as it becomes more pervasive, trace will have an even larger effect on the efficient development of embedded systems.

Michael Lindahl is a senior software engineer at Green Hills Software, 30 West Sola St., Santa Barbara, CA 93101; (805) 965-6044. Michael can be contacted at mlindahl@ghs.com



Reprinted from ECN February 2005 by Valeo Intellectual Property Inc..
Copyright © Reed Business Information, a division of Reed Elsevier, Inc. All rights reserved.
For reorders call Valeo IP 651.415.2300. For subscription information call 973.292.0783.

500387



Green Hills®
• S O F T W A R E , I N C . •
www.ghs.com ▲ 805-965-6044