Simulation Fulfills Its Promise for Enhancing Debug and Analysis

Lori Vidra

ABSTRACT

In the past, simulators did not prove realistic or feasible for debug and performance analysis in the early stages of development. Now, however, advancements in fast simulation technology and analysis tools have enabled developers to speed up the development cycle by allowing them to evaluate system alternatives more effectively. This article will discuss the simulation enhancements that make this possible.

Contents

1 Introduction ................................................................. 1
2 Advantages of Simulation .................................................. 2
3 Types of Simulation Support Available .................................. 4
4 Summary ........................................................................ 6

List of Figures

Figure 1. Graphical Analysis of Cache Hits, Misses and Source ........................................ 4
Figure 2. Graphic Analysis of Repeated Cache Tuning ....................................................... 5

1 Introduction

In the ever-growing market of Internet-era embedded applications, the winners are determined by those who offer the best set of customer-desired features in the most efficient combination of hardware and software solutions. The increasing complexity of these designs, coupled with the market-driven shortened development time, require more powerful software development methodologies and tools.

In the past, developers concentrated on application analysis and performance tuning in the latter part of the development cycle once they got their hands on real hardware prototypes. Developers rarely relied on simulation tools beyond basic algorithm validation to help them in the early stages of development. Typically, developers wrote their code and used the simulator to prove software routines, check that the software was running properly, and to transition the code to a target platform. Then, much further along in the design cycle, they would move on to hardware debug, at which point the code could actually interact with peripherals.

Trademarks are the property of their respective owners.
Until now, simulators didn’t provide the developer with the needed visibility, reliability, and speeds for realistic debug and performance analysis, but times have changed. With the advancements in fast simulation technology and analysis tools, embedded software developers can now spend their time and effort in the early stages of their design to meet their unique applications requirements and reduce time-to-market windows substantially. Waiting for a hardware prototype to begin application tuning has become a thing of the past. Today DSP vendors are offering more advanced simulation solutions than available ever before, providing more extensive, in-depth visibility that can help system designers evaluate system alternatives effectively and speed up the development cycle.

Because embedded applications have become so complex and time-to-market demands so unyielding, it is essential the designers leverage the simulator capabilities available, if they are going to continue to deliver competitive, high-performance products at a reasonable cost. This article will focus on the advancements in simulators and how they can be beneficial in all stages of the development flow.

2 Advantages of Simulation

Unlike their counterparts in the mainstream processor world, DSP simulators have always had limitations that tended to overwhelm their many strengths. Perhaps most obvious, is that they have run several orders of magnitude slower than the systems they mimic. DSP tool vendors have struggled with the trade-offs between accurately modeling the device and its peripherals register-for-register, at the cost of slowing down the simulation, forcing tool developers to offer different flavors of simulators, such as faster instruction-accurate and the slower cycle-accurate versions. The more complex the task asked of the simulator, the more time required to obtain results. In fact, it was not uncommon for developers to set up a simulation and let the application run overnight to view the results in the morning, an ineffective and unrealistic approach to debugging. It is easy to see that, as applications become more complex and processing speeds faster, these simulators become less and less useful.

Another constraint of DSP simulators has been the limited amounts of data that one could collect and its inability to process and intuitively visualize the collected data. In effect, a simulator is only as good as its ability to use the collected data to enable analysis that the developer can use to optimize the code. In the past, most simulators simply could not do this, or they provided so little analysis on such a small amount of data that it provided little insight into the application it was mimicking. Thus simulators were again relegated to comparatively few of their potential applications.

Today’s DSP simulators can be used in the earliest design stages of the design and throughout the debugging and tuning process because of new methodologies that have made simulators faster and added the ability to collect and visualize data, which is often not observable in hardware. This not only makes the simulators a viable platform for development, but at times a necessity to ensuring better utilization of the full capacity of the architecture.

In improving the performance of simulators the tool vendors are making smart use of known techniques. In simple terms, simulators are redesigned to exploit the repetitive nature (loops) of DSP code, to decode the target code once and execute it multiple times. They are also using higher levels of abstractions to avoid modeling parts of the target not directly visible to the user, such as the data path of caches. The tool developer can also utilize some specific coding techniques to take advantage of the efficiencies possible in simulation.
This technique speeds up the simulator in order of magnitude. To take advantage of those speeds and make simulation a practical part of development, simulators need the ability to collect massive amounts of data. Today’s solutions incorporate a data collection capability that, while minimally impacting the performance of the simulators, can collect substantial amounts of user-defined data for additional analysis. Tools vendors are creating new analysis tools to help developers manipulate and visualize common application bottlenecks through graphical interfaces. Combining new simulator techniques, efficient software coding and data collection capability have given simulation new life in the product development process.

To perform meaningful data collections on the simulators, one should run more than core algorithms and for this, the system modeled should include more than just the core processor. This requires the modeling of cache effects, DMA, internal and external memories and peripherals. It is also important that the simulators are verified not only for functionality at the device level, but also for accuracy.

Simulators are easy to use, and as a software-based product, they require only one workstation, which also reduces cost of use. They are also highly repeatable, since simulators can endlessly run the same algorithm in exactly the same way as the developer makes needed tweaks and modifications. This distinguishes simulation from emulation because external events like interrupts are almost impossible to precisely repeat with hardware.

Simulators work by creating virtual prototypes, therefore nearly any type of configuration can be modeled, and its characteristics analyzed. As configurations are modified and run over and over on the simulator, it is possible to determine the solution that offers the best combination of performance that is suitable for the intended application. Because simulators are virtual iterations of a prototype, they provide a level of observability that makes them unique among analysis techniques. Simply defined, observability is the ability to record data and analyze sections and components of an application running on a target. From these sojourns into the circuit architecture, huge amounts of information can be amassed and used by analysis tools within the tool suite to guide the developer’s design and application development choices. This level of detail and analysis simply cannot be obtained by any other method at such low cost. Since simulation speeds have increased so rapidly in the recent past, all of this can now be accomplished quickly enough to make them truly useful.

Another benefit of incorporating simulators in the design process is that algorithm development can be performed on a CPU-only simulator, while ignoring device effects. Later, once the algorithm is optimized for its intended CPU architecture, the simulator can again be used to achieve optimization with the device effects included to ensure proper memory placement. Simulators also enable visibility into application behavior and resource usage; so that any needed adjustments can be determined up to six months in advance of prototyped hardware, all saving months of development time.

Initially, DSP simulators might appear to encroach on the capabilities of emulators, but in reality they are complimentary. In practice, however, hardware and software development typically proceed in parallel, and simulation tools are able to streamline the transition from virtual prototyping to hardware availability because they are able to more accurately model what the hardware will look like and perform, prior to the cost of silicon production.
When hardware does become available, the transition to it is smoother and faster because considerable development has been conducted early in the process, so it only needs to be refined instead of radically changed as might be the case without simulators. In short, software simulation and hardware emulation together deliver advantages never before available in the DSP world.

3 Types of Simulation Support Available

Silicon vendors such as Texas Instruments, have been working to speed simulation capabilities for the benefits of developers. The approach each company is taking is different, but it substantiates the need and importance of simulation in the design cycle.

The DSP Analysis Tool Kit recently introduced by TI as part of its Code Composer Studio™ Integrated Development Environment is a good example of how these analysis capabilities can be used to optimize performance. The components of the Analysis Tool Kit include an on-chip cache analyzer, pipeline stall analyzer, code coverage analyzer, and multi-event function profiler, all of which work in cohesion to bring the user simulation’s highly touted capabilities.

Specifically, the cache analyzer gives the user a graphical representation of cache accesses, and highlights cache hit and miss patterns over time. This tool automatically gathers the cache data from the simulation run and graphically displays the cache activity; cache hits, misses and its source, so that the developer can then isolate and identify patterns as shown in Figure 1.

![Figure 1. Graphical Analysis of Cache Hits, Misses and Source](image)

The developer can now tune the code minimizing cache misses to obtain the best performance, and then repeat the analysis, as shown in Figure 2, until the results are satisfactory.
Close analysis of a pipeline structure is often necessary to eliminate bottlenecks and maximize the efficiency of the application. Stalls within the pipeline can cause many additional cycles to be consumed without executing the application and they can create conflicts between different instructions attempting to utilize the same resource. The pipeline stall analyzer within the Analysis Tool Kit shows what is in the pipeline down to the instruction level, along with how many stalls are occurring throughout the application, their types, and the kinds of conflicts that are causing them to occur. The pipeline stall analyzer automates the entire process that some developers are already familiar with, which includes writing the instructions down, analyzing the resources used, stepping through the application to see how the CPU executed the instructions, and attempting to detect the stalls intuitively.

Another challenge for developers is determining whether or not all of the code in a test sequence is executed and/or whether unnecessary code is present in the application. To find out this information, typically the developer must instrument the source code with printf statements to truly verify that a code segment has executed. Unfortunately, this methodology affects the way the code behaves and potentially invalidates the test. The affect on performance can be especially significant in real-time DSP systems, which are designed to be as efficient as possible. A code coverage analyzer automatically finds conditional statements in the code, tracks the path taken through them, and delivers results to the developer in a graphical format. Since the analyzer methodically evaluates each line of code, it is possible to be virtually certain of code coverage without disrupting its execution.
Another “coding and tuning” challenge confronted in the simulation process is assembling the enormous amounts of information required on all aspects of application performance and presenting it in a useful way. While earlier versions of simulators made it possible to gather most of this information, including cache and pipeline activity, their slow speed made the task cumbersome. TI has taken advantage of these “turbo-charged” simulators with massive data collection capabilities and made it possible to collect data on all events simultaneously and present the information in a single, sortable table to the developer. This capability alone saves countless hours of work, and since it is performed quickly, it allows the user to tweak his code and rerun the analysis to see how changes made to one event affect all other events. The multi-event profiler stores the data so that the results of multiple runs can be compared to allow the developer to refine the code and repeat the process until satisfied with the application performance.

4 Summary

As the demands on DSP developers continue to increase, it is obvious that development tools are becoming an increasingly important asset to accelerating the design cycle. While the DSP community has functioned admirably with slower simulation methods and manual evaluation techniques, clearly the recent advances in these capabilities warrant reconsideration. The latest tools provide capabilities to ensure that development of even the most daunting applications can be performed quickly and cost effectively, and those developers that neglect to educate themselves about these new tools will find themselves continuing to struggle to get products to market in a timely manner.
IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

<table>
<thead>
<tr>
<th>Products</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amplifiers</td>
<td>Audio</td>
</tr>
<tr>
<td>Data Converters</td>
<td>Automotive</td>
</tr>
<tr>
<td>DSP</td>
<td>Broadband</td>
</tr>
<tr>
<td>Interface</td>
<td>Digital Control</td>
</tr>
<tr>
<td>Logic</td>
<td>Military</td>
</tr>
<tr>
<td>Power Mgmt</td>
<td>Optical Networking</td>
</tr>
<tr>
<td>Microcontrollers</td>
<td>Security</td>
</tr>
<tr>
<td></td>
<td>Telephony</td>
</tr>
<tr>
<td></td>
<td>Video &amp; Imaging</td>
</tr>
<tr>
<td></td>
<td>Wireless</td>
</tr>
</tbody>
</table>

Mailing Address: Texas Instruments
Post Office Box 655303 Dallas, Texas 75265

Copyright © 2004, Texas Instruments Incorporated