Taking Co-verification To the Limit
ARM Creates a Virtual Prototype of its Latest Core Running Windows CE

By
Russell Klein, Mentor Graphics
Jon Connell, ARM Ltd.
Allan Skillman, ARM Ltd.
David Streams, Microsoft Corp.

When one thinks of co-verification, the image is of a team designing a system testing out its software on a virtual prototype of their hardware. So at first glance, a world-class designer of microprocessor cores like ARM would not seem a likely candidate for co-verification. But ARM thought differently. They saw the tremendous potential for co-verification in their design process and decided to use that in their latest processor design.

Instead of verifying custom software, ARM wanted to use co-verification to validate a new embedded core's ability to support Microsoft's Windows CE operating system before actually fabricating the core. This effort would take co-verification as far up the design cycle as possible, making this a highly interesting and challenging situation for all involved—ARM, Microsoft and Mentor Graphics (the EDA supplier of choice for the co-verification environment.) Theoretically, there was no reason why this should not work. It was just that no one had ever attempted this before.

ARM9TDMI and Microsoft CE

It was important to prove that ARM's newest processors would work with Windows CE before committing to silicon. The ARM920T was designed from the ground up to support the Windows CE operating system from Microsoft. It was important to verify that the processor was backward compatible with the programs developed for the previous family of ARM processors, the ARM7 family. Supporting windows CE, compatibility with exiting Windows CE applications and hardware designs was also important. Further complicating the matter is the fact that ARM does not produce any devices itself, but instead licenses its designs as intellectual property to others. Getting a physical prototype for verification purposes has always been a complicated, time-consuming task. All these factors made co-verification an extremely compelling alternative. Using a virtual prototype environment ARM could verify as soon as possible the ability of the ARM920T to fully support the Windows CE operating system.

This processor is the third in the recently introduced ARM9TDMI Family. The ARM9TDMI cores are small, high performance, power-efficient 32-bit RISC processors that deliver a performance of 133MIPS at 120MHz on commodity 0.35µm CMOS, and over 200MHz (220MIPS) on leading edge 0.25µm and 0.18µm processes. Each core features a five-stage pipeline, Harvard buses, Thumb extension and full debug access to all programmer's model states. The Thumb code compression extension delivers 32-bit RISC performance at 16-bit system costs through the efficient use of a second, compressed set of 16-bit instructions, which reduces memory use by a third. These microprocessors are targeted at PDA's, smart phones or Internet TV's, and peripherals that require high data throughput such as an MPEG decoder in a digital set-top box.

Microsoft's Windows CE operating system is positioned to dominate this segment of the market, so the ARM9TDMI Family includes specific support for that operating system. In fact, the ARM920T's MMU is specifically tailored to support the Windows CE operating system. The performance of the ARM920T makes it suitable for a complex operating system like Windows CE, and the low power consumption make it ideal for portable, battery powered devices that Windows CE is targeting.

Microsoft Windows CE is enabling new categories of non-PC business and consumer devices that can communicate with each other, share information with Windows-based PCs, and connect to the Internet. A compact and portable operating system, it was specifically designed to support the development of a broad range of business and consumer devices. The Windows CE operating system is a 32-bit, multitasking, multithreaded operating system that has an open architecture design, providing support for a variety of devices. It is compact, providing high performance in low-memory conditions and is scalable and has integrated power management, enabling long battery life on mobile devices. It ensures predictable, bounded latencies and supports a broad range of powerful, 32-bit microprocessors including ones from ARM.

Creating the Virtual Prototype

Evaluating how well the ARM920T worked with the Windows CE operating system was vital, being that the processor core was specifically tailored for this operating system. Simply simulating the Windows CE using an instruction set simulator for the ARM920T would not be enough, because crucial hardware dependencies would not be taken into account. But using a virtual prototype meant moving into uncharted territory.

However, there was no real reason why it could not work for this type of verification. Co-verification closely weaves together the debug and development environments for the hardware and software. It carefully synchronizes the software and hardware simulators, providing system-wide debug features such as setting breakpoints in both the software and hardware domains. Just as importantly, it sustains execution speeds at least a thousand times faster than traditional simulation products to deliver the requisite throughput needed to support software execution.

ARM was determined to leverage this powerful simulation capability to carefully examine the performance of the ARM920T operating in a typical target environment. But creating a virtual prototype was a big challenge and obstacle to using co-verification for this particular application. Since this core is intended for use in a wide variety of applications—anything from palm-top computers to DVD players—that made it tough to create an adequately representative hardware model. Fortunately, Microsoft had a reference design available in VHDL. It is a single board computer that can be configured with FPGAs to support any of the processors that Windows CE supports. Called the ODO board (after the shape shifting character in Star Trek), this HDL description of a single board computer, comes complete with an LCD screen, keyboard, memory, memory interface, and a place holder for the microprocessor of choice. For this particular task, the ODO boards was configured to include the ARM920T. Microsoft had a design working with an ARM720T and StrongARM. Some minor modifications were made to the design to support the ARM920T. They also had to replace the VHDL memory models with Seamless memory models to gain visibility into these memories from the debugger. These were the only changes needed to prepare the design for co-verification. There were no changes made to the software, they ran the same Windows CE that you will one day run on your car's computer. The only issue with the software was the format of the Windows CE binary image. Microsoft's Developers Studio creates a binary image that was not compatible with the Seamless debugger, but a simple "perl" script converted it to a compatible binary image.

Once the virtual hardware environment was established, the next concern was whether the co-verification environment would be able to handle the substantial amount of code needed run the Windows CE operating system. Loading the operating system was the moment of truth for all involved. No one was quite sure how the co-verification environment would respond to this large quantity of software. There was the very real possibility that the operating system would be such a burden, that it would slow the simulation to a painful crawl, rendering the co-verification effort relatively useless.

The team, however, was relatively confident that their tool was up to the task. The reason is that the Seamless CVE can easily handle large amounts of code due to a number of patented optimization techniques that speed co-simulation by reducing the load of software operations on the logic simulator. One such innovation is a memory image server that can process general system operations—such as instruction fetch and memory read/writes—10,000 times faster than a logic simulator. As a result, designers can trade off between resolution and performance without sacrificing accuracy. The tool does this by maintaining accurate time synchronization and data coherence between the two domains, while reducing the activity that flows between them.

Co-verifying the Core

Knowing they were pushing the limits of the co-verification paradigm, the team was very pleased to find that they were able to get the co-verification environment up and running in just four days. This meant the design team at ARM could focus on examining the ARM920T running the Windows CE operating system much earlier than what they anticipated. And examine they did.

Over the next two months, the team closely scrutinizes every aspect of the core's operation. And the processor basically performed as expected, with a few exceptions. For example, as mentioned earlier the ARM920T needed to be software compatible with the ARM7 family of processors. During co-verification, the designers discovered that the interrupts were being processed slightly differently in the new core design. The new five-stage pipeline had more instructions partially executed when an exception or interrupt occurred. The partially executed instructions need to be un-done, or “rolled back” when an exception is processed. The difference in how the exceptions are processed showed up during the initialization of Windows’ virtual memory page table, which makes liberal use of memory exceptions. What impressed them was how quickly they could identify and resolve an issue like this in the co-verification environment. Uncovering this type of inconsistency between the software functionality and the logic design during physically prototype would have seriously impeded the verification process and taken possibly taken days or even weeks to find and fix.

The only reason the ARM designers were able find this problem was because they could view all the operations of the processor’s activity—giving them much more insight into the system’s behavior than what would have been observed during actual physical verification. The Seamless CVE environment gives an unprecedented view of the actual instruction execution during the logic simulation, when the user so desires. With this tool, the ARM team was able choose when to have that level of detail and when to opt for greater simulation throughput, dynamically adjusting the balance between detail and performance during the co-simulation run. That way the team could quickly move forward to those areas that interested them. Then, after examining in details, say in the device drivers, they could quickly move to the next area of concern.

Besides giving the team the ability to examine the hardware design in operation, co-verification also enabled the team to verify and debug the ARM920T model—the actual co-verification model of the processor. Because ARM is exclusively in the IP business, licensing out its processor cores to major ASIC houses and electronic manufacturers, it is extremely important that their simulation files be as accurate and robust as possible. That fact that co-verification helped them to not only validate the actual core design, but also the co-verification model, was big plus to ARM.

Proof Positive for Co-verification

There is no doubt that co-verification was a big benefit to the overall design of the ARM920T. By using a virtual prototype, the team was able to verify the most important aspects of the this processor’s functionality—namely, its ability to support the Windows CE operating system and its backwards compatibility with the ARM7 family of cores. The fact that they were able to do this before committing to an expensive silicon prototype, gave ARM a much higher degree of confidence as they moved forward towards design completion.

This case demonstrates the validity of co-verification for core designers, but what does this mean for mainstream system designers? Well, if co-verification can perform in this extreme case then it certainly is a viable approach for embedded system designers. Faced with increased functionality, most embedded design are now using advanced, sophisticated operating systems like Windows CE. The fact that ARM’s co-verification effort easily dealt with this combination of complex hardware and substantial software, proves that co-verification now delivers the performance needed to simulate today's demanding designs. Moreover, this effort also shows that co-verification is a powerful way to examine and optimize a design's backward compatibility—an important concern for most system designers.