print | close

The Melting of the ICE Age

Electronic Design

Lauro Rizzatti

Wed, 2015-02-11 09:56

Hardware emulation is the only verification tool that can be deployed in more than one mode of operation. In fact, it can be used in four main modes, with some of them combined for added flexibility. The deployment modes are characterized by the type of stimulus applied to the design under test (DUT) mapped inside the emulator, and the processing of the DUT response. They include:

1. *In-circuit emulation (ICE) mode:* A physical target system driving the DUT in the emulator via speed adapters.

2. *Transaction-based emulation or acceleration mode:* A virtual target system driving the DUT in the emulator via verification IP.

3. *Simulation testbench acceleration mode:* A register-transfer-level (RTL) testbench driving the DUT in the emulator via a program language interface (PLI) .

4. *Embedded software acceleration mode:* Software code executing on the DUT processor in the emulator.

It is plausible to mix some of the modes, such as processing embedded software together with a virtual testbench driving the DUT via verification IP.

ICE

Historically, the ICE mode was the first to be conceived, and it was the reason for devising emulation three decades ago. In ICE mode, the DUT is mapped inside the emulator that connects to the target system where the actual chip will rest.

ICE has two key advantages. First, the target system generates real traffic in and out of the DUT, essentially replacing the traditional software-based testbench. As a result, there's no need to create a testbench, removing a time-consuming and error-prone task, alleviating the burden on the verification engineering team already overstretched by testing huge designs.

Related

Hardware Emulation: A Weapon of Mass Verification

Point/Counterpoint: Hardware Emulation's Versatility

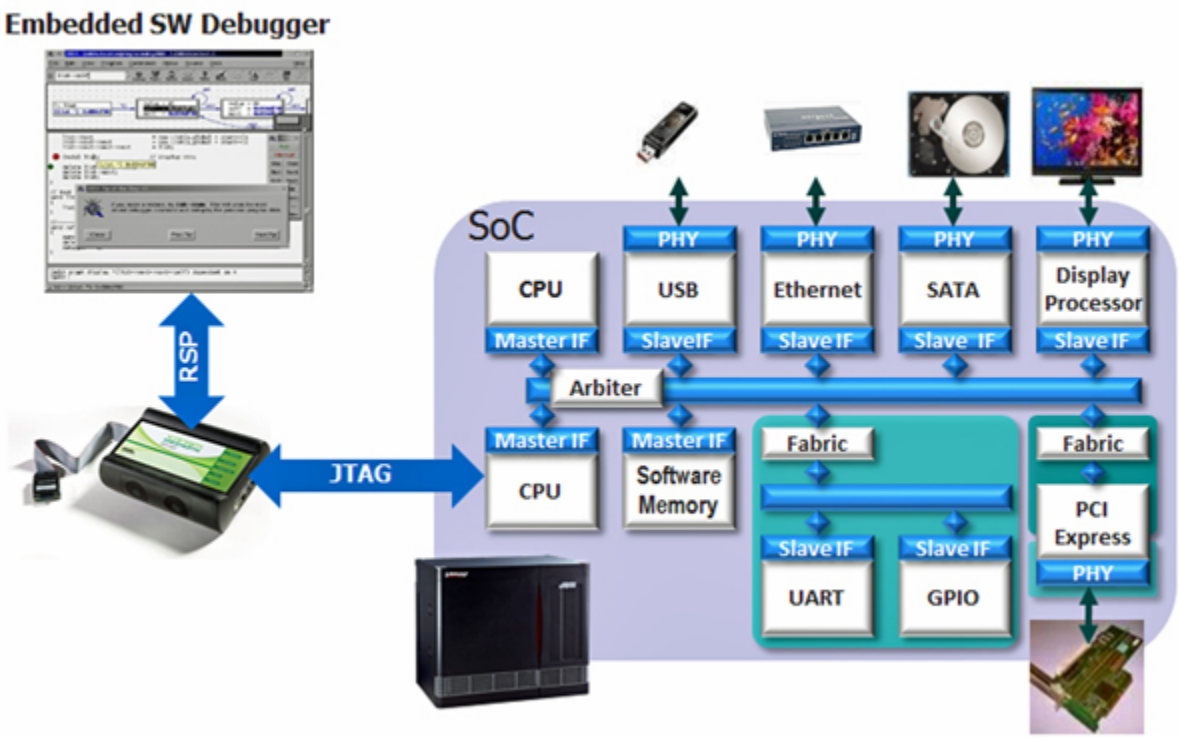
Transaction-Based Emulation Helps Tame SoC Verification

The second advantage is the ability to run at the maximum speed of the emulator, which is five to six orders of

gnitude faster than the traditional hardware-description-language (HDL) simulator. That is, megahertz sus tens of hertz. Today, this may not be true any longer.

Several issues beleaguer the ICE mode, some clearly understood and others less obvious. First, a direct connection to a target system is not possible, because its clocks run orders of magnitude faster than the clocks of the DUT inside the emulator, limited to one or very few megahertz.

To resolve this problem, speed adapters—basically, protocol-dependent interfaces based on first-in, first-out (FIFO) registers—must be inserted between the target system and the DUT mapped inside the emulator *(Fig. 1)*. This will accommodate the target system's fast clock speed to the emulator's relatively slow clock speed. The approach has shortcomings since the adapter typically trades functionality and accuracy for performance. A high-speed protocol such as PCI Express (PCIe) or Ethernet would be diminished to cope with the intrinsic capacity limitations of the FIFO in the adapter.



Moreover, the adapter's cycle-accurate behavior will differ from the actual protocol for the same reason. The setting is limited to only one test case per protocol, too, and does not allow for testing of corner cases or for any kind of "what-if" analysis. Not to mention that speed the adapters, being hardware, are subject to electrical and magnetic effects. They necessitate resetting circuitry, require magnetic shielding, and implicate cables and connectors, all elements that may make them inflexible and affect their reliability. On top of that, they have lower margins than software solutions.

Another significant disadvantage is the lack of deterministic behavior or non-repetitive behavior of the physical target system when the target system clocks the DUT. When the same test is run multiple times, it will not be identical from a cycle and timing perspective. That is, a design bug would show its effects at different points in time. Even worse, the bug may completely disappear, undermining efforts at narrowing down its location. Furthermore, the ICE mode subverts the ability to perform save and restore with physical targets. Emulation can only save the state of the DUT, not that of the target system.

The biggest drawback, though, is that it's impossible to remotely access an ICE mode setting without onsite help

plug/unplug the target system with the associated speed adapters for every user and every design—an utterly onvenient prospect.

All undermine the ICE mode and call for an alternate approach. One way is to replace the physical test environment with a virtual test environment that connects to the DUT via a transactional interface. This deployment mode is often called transaction-based verification, or TBX, for testbench acceleration, popularized by Mentor Graphics.

Transaction-Based Verification

Transaction-based verification isn't a new concept. It has been practiced for several years in the simulation world, mostly driven by the need to generate comprehensive test sequences to cope with the unrelenting increase of system-on-chip (SoC) complexities.

Conceptually, the idea is simple. Tests are written at high levels of abstraction in one of the high-level languages, such as SystemVerilog, SystemC, or C++, and the conversion from high-level commands to bit-level signals is moved into a dedicated entity called a verification IP, from here on out referred to as verification IP.

Basically, verification IP comprises synthesizable state-machine functional blocks that implement interface protocols. This allows designers to focus on creating complex testbenches, and ignore the irksome details. By raising the level of abstraction of physical interfaces from the cycle/bit-level to the transaction level, verification IP can simplify the description of data exchange between the SoC and its peripherals. Once created, verification IP can be reused. Examples of verification IP include communication protocols such as Ethernet, USB, PCIe, memory accesses, JTAG ports, or even digital cameras and LCDs.

It all sounds good, but there's a problem. The approach generates a massive amount of test cycles that bog down the simulator. After all, the simulator still executes the conversion from high-level commands into bit-level signals performed by the verification IP. It's not uncommon to run a profiler and discover that the testbench consumes more than 50% of the execution time and maybe even up to 95%, with the remainder used by the DUT.

Hardware emulation platforms have come to the rescue, making it possible to achieve a megahertz performance boost, in the same ballpark of ICE and sometimes even faster than ICE. This is done by mapping the back-end section of the verification IP that converts high-level commands into bit-level signals onto the hardware platform.

Testbenches written in SystemVerilog, SystemC, or C++ drive the front-end section of the verification IP, offering a speedup of five or six orders of magnitude compared with transaction-based simulation. It's now possible to build a virtual test environment instead of an ICE testbed by replacing a set of speed adapters (Ethernet, PCI, or USB) with an equivalent set of verification IP.

Control through Software

Unlike ICE, verification IP benefits from software control. It allows a smart verification environment to be built with a level of flexibility, and enables its reuse. For example, a designer can capture and replay traffic from live hosts; mimic in-circuit connections by forwarding transactions between the device and the chip; or apply transformations to the transaction stream to exercise hard-to-reach corner cases.

A case in point: With a PCI verification IP, it's possible to plug in a PCI software driver to an emulated design that includes a PCI interface, similar to what can be done via a hardware speed adapter in ICE. But now a software debugger can be connected to an emulator via a JTAG verification IP and run in step-by-step mode—not possible with a JTAG physical connection.

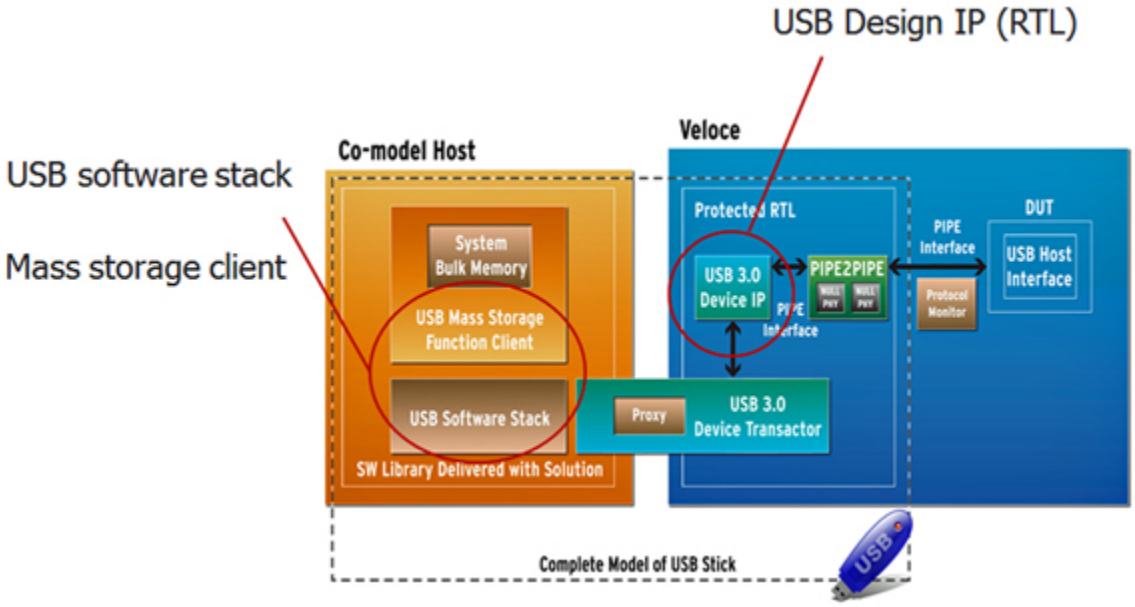
ıally important, debugging becomes easier and more efficient when designers get full control of the design :ks that aren't sourced by the hardware testbed. By controlling the clock frequency, it's possible to stop the model, read the memory content, force some registers, or dump the waveforms.

Debugging in an ICE mode requires hardware logic analyzers. In the ICE approach, designers must tolerate the whims of real hardware behavior, sacrificing repeatability. Conversely, a virtual test environment based on verification IP lets designers quickly reproduce a problem. The freedom from connecting the DUT to a hardware testbed allows designers to execute the DUT from remote locations, opening the door to the creation of emulation data centers via remote access.

It seems that testbench creation is the only remaining weakness of the virtual testbench approach. This has been addressed by Mentor Graphics with its VirtuaLAB, a physical lab inclusive of all peripherals implemented in software that can be controlled by a workstation.

Virtual Devices

Functionally equivalent to physical peripherals, virtual devices are not burdened by hardware dependencies, such as reset circuitry, cables, connectors, magnetic shields, and associated headaches. They are based on throroughly tested software IP that communicates with the protocol-specific RTL design IP embedded in the DUT mapped onto the emulator.



Virtual devices include software stacks running on the workstation connected to the emulator via a TBX link *(Fig. 2)*. The entire package allows for testing IP at the device level, and DUT verification with realistic software or the device driver itself.

Compared to hardware ICE targets, virtual devices offer several advantages:

• Easy remote access because they do not require installation of any hardware connected to the emulator.

• Flexibility for sharing a single emulator among multiple design teams because there are no speed adapters and cables to connect.

• Visibility over the target protocol software stack running on the controlling workstation.

› table summarizes the advantages and drawbacks of ICE compared to transaction-based acceleration.

IN-CIRCUIT EMULATION VS. TRANSACTION-BASED ACCELERATION		
Criteria	In-circuit emulation	Transaction-based acceleration
Test environment	No need to create testbenches — the testbench is the target system	Necessary to create testbenches VirtuaLAB removes the burden of creating testbenches
Speed of execution	Fastest speed achievable in emulation	Can be as fast as, or even faster than, ICE mode
Interface to test environment	Requires a physical speed adapter to target system for each interface Significant H/W dependencies	No need for speed adapters, but requires verification IP for each protocol No H/W dependencies
Flexibility	Inflexible, difficult to modify Potentially unreliable	Flexible, easier to modify S/W than H/W No reliability issues
Cost	Cost to design speed adapters	No H/W cost, but cost for in-house creation of verification IP or to purchase them from EDA/IP vendors
Design debug and analysis	Non-repetitive behavior results in unpredictable design bug behavior No corner cases analysis No Save/Restore capability	Deterministic, repetitive behavior facilitates and accelerates bug detection "What-if" and corner-cases analysis possible Save/Restore capability
Remote access	Remote access requires human presence	Unmanned, ideal solution for remote access

Many designers still use ICE out of habit and/or for not being aware of a better alternative. In fact, all vendors of emulation platforms claim to support ICE, albeit some implementations work better than others. But a designer's verification perspective changes once he or she experiences transaction-based verification through emulation. The ability to quickly set up a powerful test environment unfettered by cumbersome ICE hardware means easier and more effective debugging. That's why I think the ICE Age is melting.

Source URL: http://electronicdesign.com/test-measurement/melting-ice-age