Simulation + Emulation = Verification Success

If you haven’t noticed it’s the age of the SoC, though it wasn’t always so. Consider the example of personal computing, an era quickly fading into history according to many. Once upon a time, using a computer meant sitting in front of a desktop PC powered by several chips — CPU, GPU, north/southbridge, controllers for USB and ethernet ports, and so on. Now, it’s likely that your first encounter with a computer each day is tapping the screen of a smartphone in which a single SoC handles most functions. And SoCs are proliferating far beyond smartphones and tablets into everything from gaming consoles to networking gear to cars, airplanes and satellites.

Because SoCs are generally associated with reduced system complexity and cost, and also relatively compact footprints, the monolithic chips are good news for consumers. But there is bad news, too, perhaps best expressed in an old engineering maxim that still holds true and has shown up in countless quotes though the years, including this one from Steve Jobs, in a 1998 BusinessWeek Q-and-A: “Simple can be harder than complex. You have to work hard to get your thinking clean to make it simple. But it’s worth it in the end because once you get there, you can move mountains.”

The fact is there’s little that’s simple about designing and verifying SoCs. One reason stems from the choice and flexibility — all benefits have a downside — that come with assembling the chips. In the case of ARM, for example, companies can either buy off-the-shelf processor designs made by the British company or build their own processors that run on the ARM instruction set. Then comes the task of linking these processors to other necessary components, including memory, which also can be purchased and tweaked or built from scratch. Eventually the paradox of choice kicks in — more options about the way to solve a given problem eventually can lead to engineering anxiety and almost always slows things down.

“It’s not enough to just say ‘go to the emulator,’ because too often this means making radical changes to the verification environment, a problem I’ve experienced on other projects. What we wanted to do was to take the same verification environment that we have in simulation and use it, with only small modifications, on the emulator.”

— LANFRANCO SALINARI, ST SOC VERIFICATION ENGINEER

The second reason has to do with Moore’s Law, which continues to plod ahead, seemingly oblivious to regular reports of its impending demise. According to the 2012 Wilson Research Group Functional Verification Study sponsored by Mentor Graphics, about a third of new chip designs target a feature

CUSTOMER PROFILE

STMicroelectronics is one of the world’s largest semiconductor companies with net revenues of US$ 8.49 billion in 2012. Offering one of the industry’s broadest product portfolios, ST serves customers across the spectrum of electronics applications with innovative semiconductor solutions.

Design Challenge

- Understand and benchmark a combination of ARM components at the heart of a new SoC reference design with a verification environment that links software-based simulation and hardware-based emulation in a common flow.

Solution

- Use Mentor Graphics verification IP (VIP) when building a testbench for the Questa Advanced Simulator.
- Connect this testbench to a Veloce emulation system via TestBench XPress (TBX) co-modeling software.
- Separate all blocks of design code into two domains — synthesizable code, including all RTL, for running on the emulator; and all other modules that run on the HDL portion of the environment on the simulator (which is connected to the emulator).
- Work with Mentor Graphics to fine-tune the new co-emulation verification environment, which requires that all SoC components be mapped exactly the same way in simulation and emulation.
size of 28 nm or less and contain more than 20 million gates, and 17% of all new designs are greater than 60 million gates. About 78% of all new designs have one or more embedded processors. These numbers should give some pause, particularly given the rule of thumb that verification complexity scales exponentially with gate count.

These numbers quite literally give pause and just plain slow things down when it comes to verifying designs solely on software-based simulators. Despite advances from the big EDA vendors, including Mentor, at a certain point it’s simply impractical to gather a significant amount of information about a design just by feeding RTL into a simulator. Thus the increasing appeal of hardware emulators, which can offer a 1,000x increase in performance over the best simulators.

At least historically, this performance boost came at a cost, and not just in terms of the vastly different purchase prices of the two verification tools. For years, it’s been difficult or impossible to reuse verification work across a simulator and emulator. The two environments often required different flows executed by entirely different teams. However, the recent experience of STMicroelectronics engineers in Milan, Italy (supported by colleagues in Noida, India) suggests that things might be changing — or more specifically, that it might be possible to link a testbench running on a simulator with a design executed on an emulator and thus take advantage of the best of both verification platforms.

**If you build it (a reference design), they will come**

Yes, SoCs are hot, evidenced by the fact that nearly all the big semiconductor companies are building and shipping these all-in-one chips. Among the big players: Apple, Qualcomm, Samsung, Nvidia, Texas Instruments, Intel and AMD. ST belongs on this list, as well, particularly given its success in SoCs for smart home, health and wellness, and automotive applications.

Last year the Geneva, Switzerland-based company undertook a pilot project to build what it called the Eagle Reference Design, or ERD. The goal was to see if it would be possible to stitch together three ARM products — a Cortex-A15, Cortex-7 and DMC 400 — into one highly flexible platform, one that customers might eventually be able to tweak based on nothing more than an XML description of the system. But to offer this level of customization to its customers, ST would have to make sure that the ERD was backed by an efficient verification environment, one that would create the straightest possible path from verification of RTL to software development and porting of an operating system. This meant creating an environment to make it easy to measure performance and optimize the platform.

But first things first. While a verification environment like this might someday help convince ST customers to place orders for large numbers of ERD-derived designs, at the beginning of the project the ST team needed to understand the ERD itself.

“Before we could do anything we had to be able to understand and benchmark the complete ARM subsystem, which was new to us,” says Alessandro Daolio, a hardware design project manager in Milan. “We wanted to start benchmarking it running a significant amount of software, but even simple benchmarking is impossible without the right verification environment.”

**Why it’s hard to deliver higher levels of abstraction**

Projects like the ERD are symptomatic of a trend that’s pervasive in technology. In a nutshell, the game today is to offer ever higher levels of abstraction and choice. Customers, whether teenagers buying new smartphones or engineering teams

“Our customers want assurance that we are using state of the art verification.”

— ALBERTO ALLARA, ST VERIFICATION ENGINEERING MGR
shopping around for SoCs, want more functionality, performance, and ease of use, along with lower costs. The implicit promise in a reference platform like the ERD is that it offers an architecture that’s fundamentally solid and robust while still allowing for customization. It’s not unlike car manufacturers offering buyers the ability to configure a dream vehicle through their websites.

“Our customers want assurance that we are using state of the art verification,” says Alberto Allara, an ST verification engineering manager.

Precisely what qualifies as state of the art has changed in the more than 15 years that Allara has been doing verification. He cut his teeth doing directed testing, first in Verilog and then using SystemC or Tcl-based tests on a Verilog testbench. Next came an embrace of SystemC verification components, then the increased standardization and functionality of SystemVerilog OVM and finally UVM. Now, he says, the next trend is to somehow accelerate verification. Examples include using a powerful GPU to speed up simulation or integrating an FPGA into the verification environment.

In terms of raw speed, performance, and compile times, the best option by far is to make use an emulator with a custom ASIC optimized for routability rather than an off-the-shelf FPGA in which the goal is to maximize the number of logic cells used. Mentor’s Veloce2 is based on such an ASIC. As an example, the process of getting a new model of a modified design compiled for the emulator might take hours for an FPGA but just minutes for Veloce2.

However, when you consider the components of the platform, offering anything remotely resembling point-and-click configuration of an SoC is astonishingly difficult. The dual core Cortex-A15, for example, has approximately 150 million transistors, which of course will switch on and off in different sequences as changes to the overall platform are made. Customers want to know that these changes are accounted for in verification. The coverage model, for instance, needs to adapt to the different configurations.

Any potential time savings might easily be swamped by the days or weeks required to learn how to use the emulator and to link its verification environment to the simulator’s. Several Mentor Veloce boxes have in fact been used by ST for many years. However, the emulators are all at ST’s Grenoble, France, campus, nearly 400 kilometers away from Milan. And when it came to the emulators, Allara, Daolio and their colleagues were separated by more than just distance.

Veloce2’s graphical command, control and runtime interface
Prior to this project, ST used their Veloce hardware mainly for in-circuit emulation. Groups like Allara’s would hand off a database, including RTL and a simplified testbench, to a separate team in Grenoble dedicated to emulation. This French team would, in turn, map the SoC, connect its external interfaces with real hardware, and provide a platform that a typical software developer or application engineer might use.

The potential benefits of emulation weren’t lost on those, like Lanfranco Salinari, used to doing verification strictly on software-based simulators. Salinari, an SoC verification engineer, worked with Daolio and Allara on the ERD project.

“Software-driven tests are becoming more complex; just configuring them can take a long time. So we all know it’s necessary to somehow accelerate simulation, and an emulator is potentially an attractive means to do just that,” he says. “But it’s not enough to just say ‘go to the emulator,’ because too often this means making radical changes to the verification environment, a problem I’ve experienced on other projects. What we wanted to do was to take the same verification environment that we have in simulation and use it, with only small modifications, on the emulator.”

A brief thought experiment might help illustrate just what the ST team wanted to do:

Imagine someone with a driver’s license and a decent car is looking for a way that’s faster than driving to get from point A to point B — or more specifically, from Milan to Grenoble. Sure, he could fly, though if he factored in the hassle of flying commercially, including buying a ticket, parking, going through security and waiting in other lines at the airport and so on, he’d probably find that it’s faster just to hop in his car and drive across the border into France. Now imagine it’s several decades in the future and that many inventions long imagined by science fiction authors, including flying cars, have finally come to pass. Our imaginary traveler has one such flying car and his license entitles him to take to the skies too. In this Jetsons-comes-true scenario, it’s almost certainly faster to fly.

Daolio, Allara and Salinari wanted what, until very recently, was just as farfetched as that flying car.

**How to extend simulation in a few easy steps**

The ST team had long used a simulator other than Questa. So when the team first began setting up a flow to speed things up via the Veloce emulators in Grenoble, the choice was to continue with the simulator they were familiar with as the first stop in verification.

Mentor Graphics verification IP (VIP), a library of reusable components that integrate seamlessly into testbenches built using UVM, OVM, Verilog, VHDL, and SystemC, also works well with third-party simulators. Accordingly, Daolio and his colleagues first built a complete simulator on their existing tool. This work done, they then ported the testbench to Questa, which was required to connect to Veloce via the TestBench XPress (TBX) co-modeling software application. TBX makes it easy to tap the Veloce2 transaction-level modeling (TLM) verification engine running up to 10,000x the speed of TLM software simulators.

In co-modeling, reusable testbenches are interfaced to synthesizable transactors collocated with the DUT in Veloce2. These “accelerated” transactors convert high-level transactions to signal-level stimuli to drive the DUT. The primary transaction modeling constructs of TBX are SystemVerilog DPI exported and imported tasks. This made it straightforward to use Mentor supplied VIP — the same VIP ST used in building the testbench in simulation. TBX automatically generates a direct communication interface between the SystemVerilog or SystemC environment on a host and the SoC DUT in the Veloce2 emulator. With SystemVerilog testbenches, TBX executes the testbenches using the Questa simulator on the host PC.

After some tinkering, this flow — third-party simulator to Questa to Veloce — worked quite well. However, as the project progressed and time became constrained as it inevitably does, the

**“Before we could do anything we had to be able to understand and benchmark the complete ARM subsystem, which was new to us. We wanted to start benchmarking it running a significant amount of software, but even simple benchmarking is impossible without the right verification environment.”**

— ALESSANDRO DAOLIO, ST HARDWARE DESIGN PROJECT MGR
team opted to directly build a SystemVerilog testbench for Questa. This move helped them to meet their end goal, which is slightly different than the norm because they were just working on a reference design. Despite the advantage of off-the-shelf VIP, and the advantages of integrating a simulator and emulator from the same vendor, the project still entailed a significant amount of work — evidence of the complexity of modern SoC verification. The project began in May 2012 and was nearly completed by May 2013, when this article was written. (As the ERD was just a reference design, occasionally members of the team got pulled into other projects.) Nine ST engineers were involved, including five in Milan, two in Naples, and two in Noida, India. Piyush Kumar Gupta and Sachish Dhar Dwivedi are based in Noida and part of the ST’s HEVS (Hardware Emulation and Verification Solutions) team. The two provided much assistance in mapping and setting up the design in Veloce, critical since the Italian ERD team had no experience working with emulators prior to this project.

“Because this reference design was not bound to any particular project, our main goal was not to arrive at the complete verification of the design,” says Salinari. “Above all, we wanted to be able to do performance analysis and establish the verification methodologies and techniques so that, when we eventually try this on a real project, we’ll be able to port the verification environment to the emulator more or less seamlessly.”

Basically, the task was to separate out all the blocks of code into two domains — synthesizable code, including all the RTL blocks and the SystemVerilog interface that contains the design simulation model (DSM) of the VIP, that can run on the emulator; and all the other modules that run on the HDL portion of the environment on a host machine (the simulator) connected to the emulator. The team added its own twist, too, described in a paper by Allara and his colleague Fabio Brognara and presented...
The paper describes a mechanism developed by Allara and Brognara that named the verification abstraction layer (VAL) that is used to expose the external verification environment to the software running on the system’s CPU. ST verification engineers can use VAL to program and configure not only the internal design IP but also the external verification component.

“This mechanism, which we developed for an earlier project and that works so well in simulation, was ported during the ERD project to run in emulation on Veloce,” says Allara. “We developed a front end to VAL that can be mapped on the emulator. So currently, we have software running onboard the CPU master in the emulator that can control the verification component that’s executed on the host machine running the simulator.”

As the ST team describes it, establishing this new verification environment was not a matter of solving one or two big issues but rather of dealing with many small issues. As just one example, the team had to approach clock generation in a new way.

Previously, during verification, all phase-lock loops (PLLs) were abstracted, and the clock was generated using external Tcl scripts. Co-emulation requires that all SoC components be mapped exactly the same way in simulation and emulation. The team discovered that, to keep everything aligned, it needed something more detailed than just abstracted PLLs that could work in both domains.

Moving forward with a commitment from Mentor

The project wouldn’t have worked without help from Mentor, which is working hard to support this phase of verification acceleration that will make better and more integrated use of emulators by teams that until now have done almost all their work on simulators.

Salinari says Mentor’s VIP was the essential ingredient.

“Without it, we would have had to build much complex verification-related IP and figure out how to navigate complex protocols — work that was simply out of the question because we don’t have the resources,” he says. “The VIP provided by Mentor allowed us to run the same verification on both simulation and emulation. Sure, we had to make some tweaks, but at least it was possible.”

And in all likelihood, this co-emulation will be essential moving ahead, largely because it’s a much more efficient and cost-effective way to approach verification. Ironically, one reason for this is that the emulator is used much more like a simulator. Or more specifically, it provides a better way to accelerate simulation while preserving familiar testbench architecture and methodologies. Regression tests can be run at night or on the weekend in batch mode.

Allara says he and his colleagues got much support from Mentor, including lots of technical consultation and voluminous technical documentation. More than that, he said, is the impression that Mentor is committed to help change the economics of chip and full system design.

That impression is backed up by an interview with Mentor CEO Wally Rhines that appeared in Electronics Weekly in April 2012.

“The emulator becomes the hub of the verification process,” Rhines told the U.K. based publication. “It takes on the function of the simulator and improves on it.”

“Let’s just say we’re still learning how to best use the emulator in conjunction with simulation,” says Allara. “But our feeling is that Mentor really believes in this technology and will continue to support us, and just generally is committed to flattening the learning curve for verification engineers who aren’t emulation experts.”

For the latest product information, call us or visit: w w w . m e n t o r . c o m

©2013 Mentor Graphics Corporation, all rights reserved. This document contains information that is proprietary to Mentor Graphics Corporation and may be duplicated in whole or in part by the original recipient for internal business purposes only, provided that this entire notice appears in all copies. In accepting this document, the recipient agrees to make every reasonable effort to prevent unauthorized use of this information. All trademarks mentioned in this document are the trademarks of their respective owners.