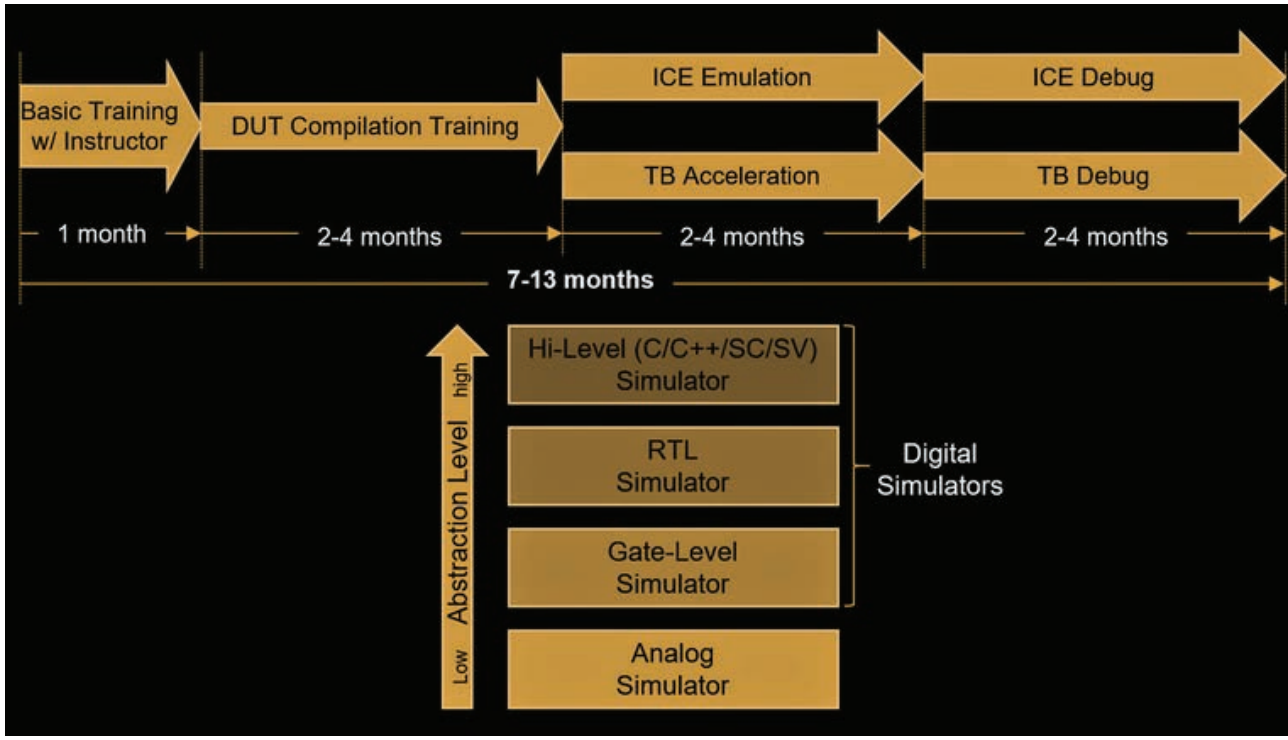
ElectronicDesign®



Test & Measurement

How to Become an RTL Simulation Expert vs. Hardware Emulation Expert

Part 2 of this two-part series on designing and using a modern hardware emulation platform highlights how to become an expert in using a modern hardware emulator vis-à-vis a hardware-description-language (HDL) simulator.

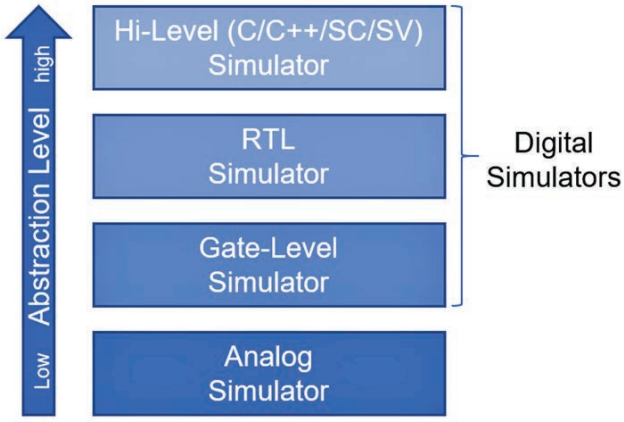
By Lauro Rizzatti

In Part 1, we reviewed the process of designing a modern hardware emulation platform. Here, we'll look at the skills and training that are necessary to become a simulation expert and an emulation expert. A good starting point is to briefly review differences between a simulator and an emulator.

Simulators

As the electronic design approach evolved over time, and in turn drove design engineers' creativity up the abstraction hierarchy, the industry invented simulators targeting a range of design-under-test (DUT) design descriptions at different hierarchical levels. At the bottom sits the analog simulator that mimics accurately the electrical behavior of transistors in the DUT.

All simulators above analog are digital. They process the DUT at increasingly higher abstraction levels than transistor analog voltages and current, from gate level to register transfer level (RTL) to higher levels represented by languages such as C/C++/SystemC and SystemVerilog. *Figure 1* maps the simulator hierarchy and the abstraction levels at which they operate.



1. The simulation hierarchy starts at the transistor level with an analog simulator. Digital simulators are used at the next level of design and above. (Source: Lauro Rizzatti)

All simulators, regardless of the type, consist of a complex software algorithm executed on a workstation or PC. Of all the types of simulators, the RTL simulator is the most popular and, indeed, the premiere tool for functional verification.

From a user perspective, the deployment of an RTL simulator proceeds in two stages. First, a compilation process converts the DUT from a source-level description into a binary database. Second, the simulator reads and evaluates that database to produce a response to a given stimulus. The compilation process is virtually automatic and doesn't require user engagement.

Stimulus creation is a topic that could be discussed on its own. It should be noted that simulators don't have the processing power to execute and validate embedded software. Their deployment is confined to verifying the design hardware.

A popular approach to stimulus generation today is based on the universal verification methodology (UVM). While knowledge of a simulator is a requirement to learn UVM, a verification engineer can learn how to use a simulator without any knowledge of UVM. To avoid learning a tool and a methodology at the same time, the novice user should focus on the tool first and create some basic stimulus manually.

Differences among commercial simulator implementations involve the graphical user interface (GUI), including waveform display and analysis, stimulus creation, the breadth of supported features and debug capabilities, and speed of execution. Occasionally, differences in the simulation algorithms may lead to discrepancies in the quality of results.

Emulators

A rather different landscape meets the observer of a hardware emulation engine. Part 1 described the inner workings of a modern emulator. To recap, a hardware emulation system sits on four technological pillars—the hardware, the compiler, the run-time and DUT debug environment, and supporting verification intellectual properties (VIP). If the emulator is using a custom-made reprogrammable device, the pillars become five.

Four or five columns encompass a hardware component, similar to a supercomputer, and an intricate design compilation process that maps the DUT into the emulator. A cutting-edge run-time operating system allows the user to carry out verification and debugging of the DUT. A library of VIP consisting of models of the DUT I/O interfaces makes it possible to monitor and control several characteristics of the traffic flowing through them.

Technological differences between simulators and emulators aren't insignificant. From a user perspective, an emulator is a more complex tool than a simulator to learn and deploy since the user must master four technological domains.

An emulator can be used in two modes: in-circuit emulation (ICE) or virtual. The ICE mode requires hardware knowledge while the virtual mode requires familiarity with C/C++ and SystemVerilog programming. Developing expertise in either mode needs months of practice.

An advantage of the ICE mode is that the DUT gets tested with real-world traffic, as opposed to synthetic stimulus created in virtual mode. An advantage of virtual mode is that synthetic traffic provides more direct control to traverse corners of test state space and reach coverage closure more efficiently.

The most popular approach to generate a stimulus describes a testbench at a high level of abstraction—using the SystemVerilog language, for instance—and connects it to the DUT in the emulator via one or more direct programming interface (DPI)-based peripheral interfaces. The approach prevents any slowdown in execution. Such interfaces, called transactors, are made of a C- or SystemVerilog-based front-end communicating with the testbench and a bus-functional model (Emulator and third-party vendors provide off-the-shelf libraries of transactors and users can expand the library designing their own custom transactors.

Modern emulators deployed in virtual mode don't require knowledge of the underlying hardware. The virtual mode shields the hardware from the user. Hardware knowledge plays a role when the emulator is used in ICE mode. The interface between the emulator and the real world in ICE mode is a complex and tricky piece of hardware susceptible to several undesirable physical effects and exposed to hardware dependencies.

The virtual mode also expands the use of emulation to

Criteria	Custom Emulator-on-Chip	Custom Processor	Commercial FPGA
DUT Compilation	Fast	Fast	Slow
Design Capacity per Box	2.5BGates [in a Monolithic Cabinet]	576MGates	2.4BGates
Execution Speed	1-2MHz	1-2MHz	3-4MHz
DUT Debug	Full DUT Visibility	Full DUT Visibility	Limited DUT Visibility
Virtual Mode	• 512 Gb/s I/O bandwidth • Large library of virtual peripherals • VirtuaLAB support	• (*) I/O bandwidth • Library of virtual peripherals	• (*) I/O bandwidth • Library of virtual peripherals
ICE Mode	• Large number of I/O pins support high I/O bandwidth • Large library of speed adapters • "Deterministic ICE" capability	• Large number of I/O pins support high I/O bandwidth • Very large library of speed bridges	• Fewer I/O pins limit I/O bandwidth • Very limited library of speed adapters

(*) No published I/O bandwidth specifications available

The table compares the main characteristics of architectures used in the three currently available commercial emulators. (Source: Lauro Rizzatti)

carry out verification tasks well beyond hardware debugging, which is the realm of the RTL simulator.

Not all commercial emulators are created equal, which complicates the comparison between simulators and emulators. As described in Part 1, three vendors control the market, each using a distinctive hardware architecture that contributes to their specifications *(see table)*.

With this in mind, let's review requirements to become an expert in each of the two verification engines.

Requirements to Become an RTL Simulation Expert

The road to becoming a simulator expert is rather straightforward. The basic requirement is to have a solid knowledge of hardware design and be able to navigate a design spec, making sense of its functionality.

First, access to a simulator online following a well-organized tutorial ought to provide the novice user a basic understanding of the GUI, simulator commands, and its main features. The tutorial should train the apprentice to debug a design using the whole set of the simulator debug capabilities. Quick access to User and Reference Guides must be part of the training session.

During the tutorial, the DUT size may be limited to between 1,000 and 10,000 gates or so. Larger designs may slow down the speed of execution, distracting the novice.

Compiling the design isn't difficult. No design preparation is required, and the process is fully automatic. As for stimulus creation, a simple and fast approach is to create directed testbenches for simplicity.

The duration of a basic online tutorial may be approximately one week, long enough to teach an engineer to use a modern simulator even without live support. Once the user acquires the basic knowledge, his or her attention can be directed to learn the UVM methodology and, in parallel, continue to use the simulator. *Figure 2* maps the timeline to becoming a simulation expert.

Requirements to Become an Emulation Expert

The path to become an emulator expert is longer and more arduous. As with simulation, the basic requirement involves possessing a solid knowledge of hardware design. The starting point is learning to compile a design for emulation by training with an instructor.

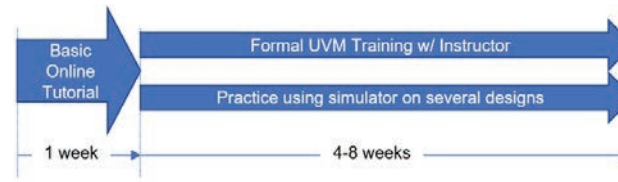
As mentioned earlier, different emulator architectures impact the compilation process, among other use aspects. Some architectures ease the DUT compilation since they allow for an automatic or almost automatic approach. Yet, all emulators require a degree of knowledge that can best be acquired via a training class with a qualified instructor. To reach a level of proficiency to successfully compile the DUT means an investment in time—from several weeks to several months depending on the emulator architecture and the DUT complexity.

In general, DUT compilation can benefit from emulator hardware knowledge. This is especially true of a field-programmable gate-array (FPGA)-based emulator. Partitioning, placing, and routing a design of more than one billion gates on an array of commercial FPGAs isn't trivial. Understanding the hardware configuration may help navigate through the gate netlist place-and-route.

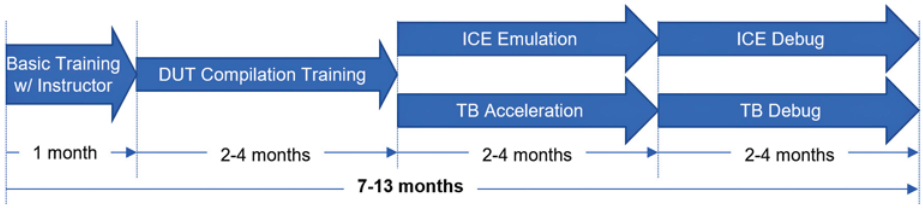
As mentioned earlier, an emulator can be used in either ICE or virtual mode. Gaining expertise in either mode requires several months of training and practice.

While ICE mode doesn't require stimulus creation since the DUT is tested with real traffic, virtual mode demands the complex task of creating a testbench. One vendor offers VirtuaLAB, which includes a complete set of software that generates and monitors application-specific traffic to/from the DUT. VirtuaLAB connects to the DUT via a transaction-based interface and can be deployed as plug & play.

DUT debugging is another area that requires compre-



2. The timeline to become a simulation expert ranges from four to eight weeks. (Source: Lauro Rizzatti)



3. The timeline to become an emulation expert takes approximately seven to 12 months. (Source: Lauro Rizzatti)

hensive training.

A simulator allows for a high degree of interactivity, including 100% design visibility and controllability, though execution slows as more of them are used. Conversely, an emulator may not offer the same level of interactivity because visibility and controllability vary depending on the architecture of the emulator. Emulators built on custom architectures achieve the highest level of visibility with the lowest speed degradation. Emulators designed on commercial FPGAs trade visibility for speed of execution.

The novice engineer must learn the emulator architecture and its limitations, and then adapt to inherent limitations. All take time to reach a level of confidence to debug a DUT.

Figure 3 maps the timeline to become an emulation expert.

Conclusion

Hardware emulation is a mandatory design-verification tool today as RTL simulation was in the past. The deployment of a simulator, specifically an RTL simulator, requires knowledge in a single technological domain. To become an expert in RTL simulation, the novice verification engineer must invest several months of training. From there, graduation with a black belt will be achieved by building experience tackling larger and more challenging designs.

Deployment of an emulator requires knowledge in four distinctive technological domains. To become an expert in emulation takes at least four times longer than mastering an RTL simulator and, in the process, demands more concentration and intellect.

Dr. Lauro Rizzatti is a verification consultant and industry expert on hardware emulation. Previously, Dr. Rizzatti held positions in management, product marketing, technical marketing and engineering.