

System architects get help from emulators

EMULATORS, WHICH DESIGNERS ONCE USED ONLY FOR DEBUGGING AT THE LAST STAGES OF DESIGN IMPLEMENTATION AND FOR REGRESSION TESTING, ARE NOW BECOMING USEFUL TOOLS FOR SYSTEM ARCHITECTS, AS WELL.

Engineers involved in designing leading-edge ICs face a number of difficult challenges. They have to deal with complex physical phenomena on die that require special DFM (design-for-manufacturing) methods and tools, and they have to handle demanding architectural and integration tasks as they plan, develop, and verify the logic design of the IC.

Consumer-electronics and communication devices are the sectors with the greatest vitality in today's electronics market. These products have an average market life of less than one year, and they compete in an aggressive, price-sensitive market; thus, profit margins are low. To be successful in these markets, companies must shorten development time and lower both development and production costs. The result is that the average development time for a leading-edge IC, or SOC (system on chip), is nine months, and its complexity is often greater than the circuits that used to take double the time to develop.

One way companies shorten the design cycle is by reusing functional blocks. This approach has given life to a new segment of the EDA industry: third-party-designed functional blocks. This market segment received the unfortunately chosen abbreviation "IP" (intellectual property), a label that more frequently means Internet Protocol.

CHALLENGES IN SOC DESIGN

Most SOC devices require engineers to integrate a microprocessor, one or more DSPs, one or more intelligent controllers, some memory, embedded software, and at least one high-speed bus on a single die. In almost all cases, third parties provide some of the functional blocks, most often in the form of hard macros. A hard macro is a black-box circuit. To protect the macro from unauthorized copying, the vendor describes the functions the circuit will perform and the I/O protocol it will use without revealing any information about the internal architecture.

These IP blocks offer a challenge to system integrators because, to protect proprietary information, vendors often provide only a TLM (transaction-level model) of the device. This model is sufficient to verify the interface protocol between the IP and the rest of the system but does not give accurate timing information and may mask errors in the actual data even if the I/O protocol seems to be correct. In addition, engineers face the possibility that the simulation model contains bugs that the vendor has been unable to identify.

When planning a new product, designers begin by listing the required functions. They then partition the product into func-

tional blocks that they will implement in either hardware or software. Engineers implement hardware blocks from scratch, reusing blocks they developed internally, or using third-party IP. The decision to use a third party requires engineers to evaluate the IP, and the process can be time-consuming because they must develop a verification and validation testbench and then evaluate the results. When the only method of evaluating an IP block is to use a simulation model, the results may not represent the actual behavior of the IP because errors can exist in both the model and the testbench.

By increasing the level of abstraction of the models, EDA vendors have given architects a new family of planning tools that allow engineers to simulate the interaction among the various system building blocks before detailed implementation begins. The early proponents of ESL (electronic-system level) predicted that, by using a language that was similar to what software engineers use, architects could more efficiently explore hardware/software trade-offs. Unfortunately, this prediction proved to be only partially true. DFM considerations often require estimating the power consumption of a software block more precisely than is possible using abstract processor models. Software engineers must be able to execute the program as early in the development cycle as possible, so the team can use accurate run-time data to decide on the best architectural configuration.

Once the architecture is final, development of both hardware and software begins. Ideally, the process is concurrent, but, in practice, software development must wait until the hardware that will execute the software is available. Some EDA vendors have offered an approach to the serial nature of this process with

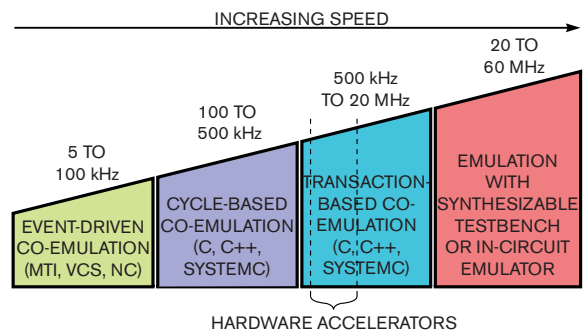


Figure 1 Using more abstract models, hardware accelerators, and emulators increases the speed of digital-logic simulation.

an ISS (instruction-set simulator) that provides a TLM of a processor. Software engineers use the model to validate the algorithmic behavior of their code, but they cannot verify real-time software using an ISS, because the model is not timing-accurate. An ISS model is expensive to build, and engineers can obtain an ISS only for popular embedded processors.

During the development process, engineers must be able to simulate the system using functional blocks at various levels of abstraction. Although they can describe some blocks only at the algorithmic level, the development of others progresses to the RTL (register-transfer level) and readies them for synthesis. Using a logic simulator to validate the design, which contains many RTL blocks, is time-consuming and expensive. If the verification of software implementations requires RTL blocks, the throughput of a logic simulator is so low as to make the approach unrealistic.

HARDWARE EMULATION IN SYSTEM DESIGN

Although ISS models, TLMs, and pure C or C++ models all provide system designers with the means to evaluate basic system architectures to partition the design, they fail to support analysis in the time domain. These models purposely lack timing models because the major directive in developing these models is to ensure the fastest possible simulation throughput. But most embedded software is time-sensitive, and communications among the various hardware blocks must involve synchronization in time. Waiting for an RTL representation of the design to verify timing is impractical for two reasons. The simulation of RTL circuits is significantly slower than using more abstract models. The resulting delay in verifying software significantly increases development time, causing greater NRE (nonrecurring-engineering) costs and possibly pushing the product beyond its market window.

EDA vendors have addressed the throughput problem by creating hardware accelerators that compile the circuit into hardware primitives. EDA companies such as Cadence (www.cadence.com), Mentor (www.mentor.com), and Tharas Systems (www.tharas.com) market hardware accelerators that provide adequate execution speed but at a significant cost. These hardware boxes can increase the gate-level throughput of a logic simulator by 1000 times, but their cost can run to millions of dollars, which makes them unaffordable to many companies.

Figure 1 shows how simulation performance increases when you apply various simulation methods. Hardware acceleration improves event-driven sim-

TODAY'S EMULATORS CAN COMMUNICATE WITH A SOFTWARE SIMULATOR AND ALLOW DESIGNERS TO USE ALL THE MODELS THAT THE SOFTWARE SIMULATOR SUPPORTS.

ulation so that the throughput is now equal to the lower band of transaction-based co-emulation. Hardware accelerators provide the best productivity improvement when the verification team uses them for regression testing.

Another approach is to use a hardware emulator. Emulators allow designers to implement a circuit using FPGA devices instead of an ASIC, thereby running simulations of the circuit at a much higher throughput than a software simulator can provide. When emulators first became available, all of the circuit had to reside in FPGAs, but today's emulators can communicate with a software simulator and allow designers to use all the models that the software simulator supports. The sophistication of emulators has also improved significantly. Figure 2 shows the architecture of the Zebu-UF, the latest emulator from EVE (www.eve-usa.com). System architects can perform a number of valuable tasks using an emulator with its capabilities.

One factor holding back the growth of the IP market is the inability of potential users to evaluate an IP core without entering into a contract with the vendor. Using a TLM of the core is inefficient. The IP vendor must develop and maintain the model. A vendor cannot distribute the model it used internally to develop the IP because the purpose of the evaluation model differs from the internally developed models. Engineers unfamiliar with the architecture of the commercially available TLM must be able to easily use it. It must be abstract enough to ensure fast simulation, and it must provide an accurate representation

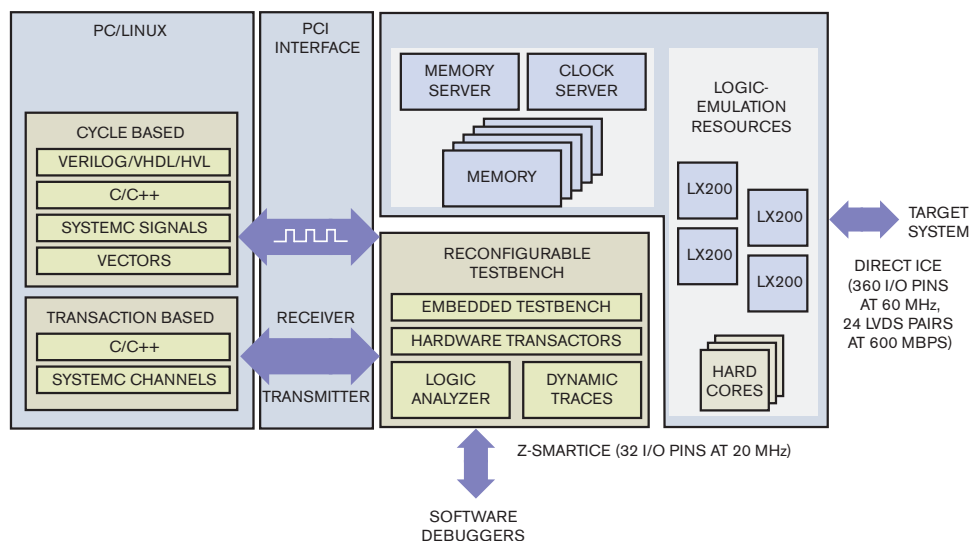


Figure 2 State-of-the-art emulator architecture supports IP evaluation, logic simulation, embedded-software debugging, and ICE functions and integrates seamlessly with traditional digital-logic simulators.

of the core's functions. The possibility of plugging a core on an emulator board and quickly executing an evaluation using the architectural description of the rest of the system provides customers with the information they need. Meanwhile, the vendor decreases its marketing cost and increases the protection of its IP.

A quarter of a century ago, a design team would not develop software for a microprocessor without using its development system. Those systems were specific to each vendor and, in most cases, to one microprocessor. The primary advantage of an emulator is its flexibility. As **Figure 2** shows, designers can connect the development environment from ARM (www.arm.com), ARC (www.arc.com), or Tensilica (www.tensilica.com) to allow software engineers to debug their software in real time and model the rest of the system at various levels of abstraction. The feature allows true parallel development of hardware and software, enabling not only a shorter development cycle, but also early trade-offs between hardware and software implementations, thereby improving the overall quality of the product.

Using a workstation and an emulator such as Zebu-UF, the development team can tailor the instruction set of a configurable processor, such as the one from Tensilica, develop and debug the embedded software, and evaluate third-party IP. It can also debug the circuitry in an environment that allows the team to have constant access to the entire design regardless of the abstraction model of each block. The ability to use various abstraction levels for models enables engineers to as long as possible defer the final decision on the architecture of the product while they are developing and finalizing some of the blocks. They can simulate the entire system, including software, with acceptable throughput speed.

MORE AT EDN.COM ▶

⊕ Go to www.edn.com/ms4160 and click on Feedback Loop to post a comment on this article.

The speed of compilation was one of the major drawbacks of early emulators. Compiling a design for emulation requires synthesis, partitioning, and a place-and-route tool (**Figure 3**). As the circuit grows, these functions grow at an increasing rate. If engineers had to wait hours between debugging runs, their productivity

would suffer significantly. Emulator and hardware-accelerator vendors can now perform incremental circuit compilation so that they need to recompile only the changes while most of the circuit remains the same.

Early access to hardware implementations is especially important to embedded-software developers. Although embedded software depends on timing and interrupts, no ESL tool can support both. Traditional programming and modeling languages, such as C, C++, SystemC, and even Java, lack support for either time or interrupt mechanisms (**Reference 1**). As a result, software engineers cannot complete their work using logic simulators; they must rely on the actual hardware. Emulators provide the earliest support for real-time software debugging.

Architects can use TLMs of the rest of the system, whereas the emulator supports gate-level implementations of portions of the system. They can study the communication pattern between various blocks, refine the partitioning, and change the

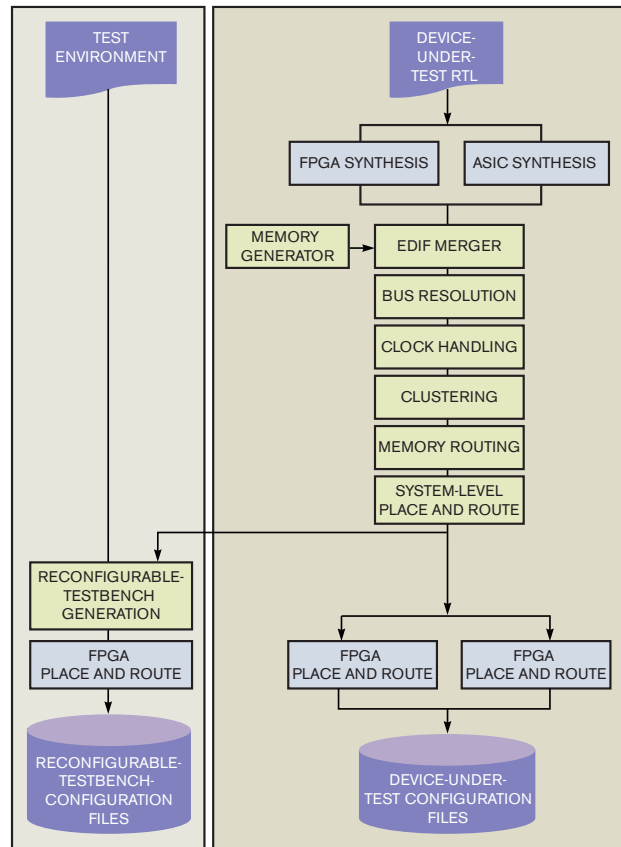


Figure 3 Engineers must map both the testbench and the design under test onto the emulator-hardware architecture. Vendors accelerate this time-consuming operation by using incremental compilation techniques.

implementation of a function from hardware to software or vice versa to minimize bus traffic. The emulator imposes no restrictions on engineers concerning the modeling language they want to use to simulate the rest of the system.

Too often, engineers see the IC as the system. Yet, every IC must reside on a board, and engineers must now consider the effects of both the IC package and the board in determining the system's parameters. Designers can use an ICE (in-circuit emulator) and observe the behavior of the entire product, not just the IC, before releasing the design to a foundry. The ICE allows architects to simulate the entire product well before finalizing the design, thus increasing the chances of avoiding a redesign. Emulators, which engineers once used only for debugging at the last stages of design implementation and regression testing, are now becoming useful tools for system architects, as well. **EDN**

REFERENCE

1 Lee, Edward E, "Absolutely Positively on Time: What Would It Take?" *IEEE Computer*, July 2005, pg 85.

AUTHOR'S BIOGRAPHY

Following a long career in engineering that culminated with his five-year stint as a technical editor at EDN, Gabe now runs GABEon-EDA, a consulting company addressing communication and marketing issues in the EDA industry. You can reach him at gmoretti@gabeoneda.com.