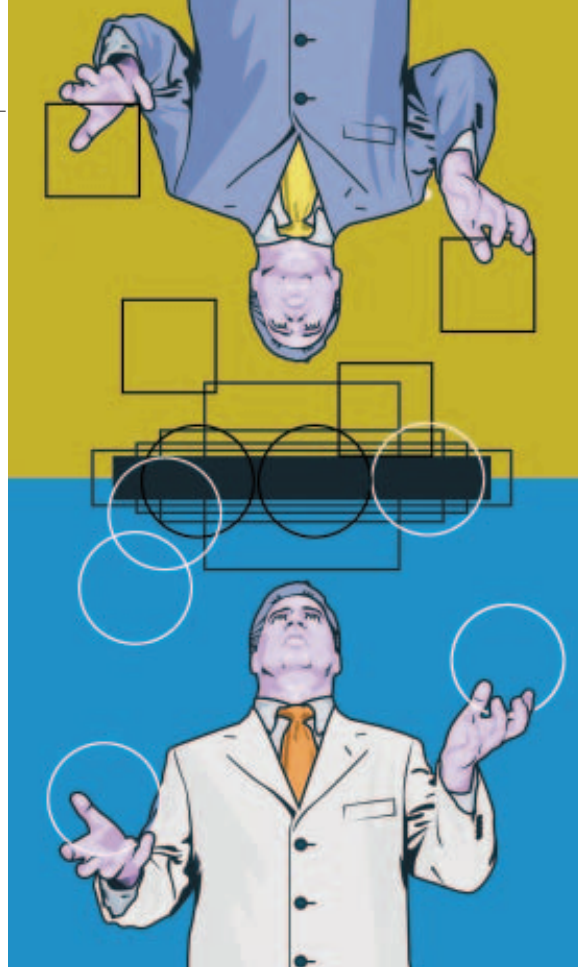


INCREASINGLY, ENGINEERS FIND THEMSELVES HAVING TO BALANCE THE CONCURRENT DEVELOPMENT OF THE HARDWARE AND SOFTWARE COMPONENTS OF A PRODUCT.

Juggling jobs



UNTIL THE USE OF ASICs became common, execution platforms were developed and debugged before most of the software for them was written. Software engineers did not complete

most of the operating-system development until the hardware was robust enough to be close to its release version. Engineers developed products using well-established and fixed platforms, ranging from mainframes to microcomputers and microcontrollers. The application software was the component that made the product unique and gave it its market superiority. The progress of the semiconductor industry in fabrication capabilities has provided engineers with millions of transistors on a die. New and improved tools from the EDA industry have given designers the ability to efficiently and effectively use the additional transistors.

An ASIC customizes the hardware architecture and, therefore, the related software, to a task. Designers create the hardware to meet the requirements of the product to be developed, often creating new computing engines and almost always designing new peripheral controllers to meet the end-product ex-

ecution-speed, power-consumption, and manufacturing-cost targets. A new hardware architecture requires new layers of software, from operating systems to device drivers and bus protocols. The operating system is rarely generic but, rather, specific to the application, and the company must modify it for every new entry in the product family it introduces.

At first blush, it may seem intuitive to first develop the hardware platform and then write the software that runs on it. This approach makes it easier for software engineers to debug the code, because they are working with a fixed execution engine and robust development tools. But sequentially developing the hardware and software presents two problems, the first of which is the length of a sequential development cycle. In many cases, it would take so long to first develop and debug the hardware and then develop the software, that the product would miss the market win-

At a glance32

The many faces of SOC verification.....32

For more information34



dow. The second problem is the potential of hardware-design errors. If you fail to detect an error you made in developing the hardware until the software is in development, the design team must go back and fix it, possibly costing them many valuable weeks or months. Further, it is likely that in the later stages of design, the fix will be more complex. The bottom line is: Concurrent development of both hardware and software components is a requirement, not a luxury.

Engineers can use a number of methods in concurrent hardware/software development (see sidebar “The many faces of SOC verification”). You can further simplify the list by condensing the approaches into three methods: hardware emulation, platform-based design, and ESL (electronic-system-level) design. The approaches continue to be hardware-centric, despite the fact that some vendors claim that their tools provide a way to evaluate trade-offs between implementing functions in hardware and implementing them in software.

The first approach gives application designers a software or a hardware mod-

AT A GLANCE

- ▶ Engineers diversify products through architecture, not just application code.
- ▶ Physical prototyping is the natural evolution of development systems and in-circuit-emulator boxes used at the beginning of the microprocessor era.
- ▶ Platform-based design simplifies the development of SOC products by providing proven hardware architectures.
- ▶ ESL offers the most modeling flexibility of all co-design approaches, but with flexibility comes complexity.

el to use while developing code before the actual hardware is available. The second method approximates the old style of having a fixed hardware target for the software to be written. Although not all of the hardware exists, most of it does, and the hardware system obeys known standards and protocols that provide a structure to the software developers. The third approach allows the greatest

amount of flexibility in the final architecture, but it is the most complex, and it requires designers to work at different levels of abstractions and to understand the limitation of the various abstraction models to make the correct assumptions.

As its name implies, physical prototyping allows engineers to use a prototype for the hardware block or blocks that are not yet available. This method has a long history and is well-understood. Years ago, when the hardware system comprised standard components, engineers would develop a breadboard that contained the known components and some hardware that simulated the digital logic not yet available to allow software engineers to debug the code before the full hardware system was available. Although today’s implementations of this method differ from the implementations engineers used 15 years ago, the approach is the same. Vendors of emulation products provide a hardware system that contains the circuitry necessary to interface with a computing system that functions as the master, the internal communication logic that allows engineers to

THE MANY FACES OF SOC VERIFICATION

By Serge Leef, Mentor Graphics

To meet the challenges and reap the rewards of SOC (system-on-chip) design, engineering teams need appropriate verification strategies that address the economic or time constraints associated with a design project. Modern verification strategies offer advantages and disadvantages.

Server-farm simulation: The server-farm approach involves distributing simulation tasks across multiple workstations. Server farms are highly capable of concurrently crunching through multiple disjointed sets of stimuli, but the approach is sometimes slow and does not enable real-time system-level debugging. It obscures the line between hardware and software by treating software as a collection of bytes stored in an HDL model of memory. This representation is clearly not conducive to

understanding the real-time behavior of software.

Hardware emulation: Emulation achieves near-real-time verification and delivers high performance and accuracy, but the cost and long setup times are prohibitive for many companies. Because you need the actual processor before you can verify a system, you cannot apply this approach to situations in which the processor does not yet exist in silicon.

ESL verification: ESL-verification approaches promise the ability to use untimed and transaction-level models of a design to verify devices at the system level. The approach promises tremendous value to the customer, but it is difficult to implement. Only a limited set of credible processor and common peripheral models exist, and those that do have no direct relationship with hardware,

because they do not model interfaces at the signal level. Although this approach is technically feasible, it will become possible only if the implementation tools and verification tools work in tandem to implement a solution.

Physical prototyping: This method involves partitioning and mapping the design into multiple FPGAs placed on a custom board. It is fast and accurate, but it is costly, and it ultimately delivers poor visibility and control. Physical prototyping is a proven but cumbersome approach that delivers limited control and visibility.

Hardware/software co-verification: Co-verification creates a virtual prototype of an embedded system months before a physical prototype is available. By executing embedded software on simulated hardware, co-verification allows designers to fully verify the

hardware/software interfaces and accelerate firmware debugging. Designers can execute boot code, hardware diagnostics, device drivers, RTOS (real-time operating-system) board-support packages, and application code on simulated hardware. By fully simulating the boot-up of a design, a co-verification tool can reduce the risk of error in the first prototype. Co-verification tools are flexible, and you can combine them with other methods to deliver even higher performance and accuracy. When coupled with emulation, co-verification can simulate the processor while the emulator simulates the SOC or ASIC hardware, or vice versa.

AUTHOR’S BIOGRAPHY
Serge Leef is the general manager of the System-on-Chip Verification Division of Mentor Graphics.

configure the hardware model of the target digital circuitry, and the mechanism required to program one or more FPGAs that model the target logic.

Charles Miller, senior vice president of marketing and business development at Aptix Corp, observes that, with an FPGA prototype running in the tens of megahertz, the operating-system boot time is only a few minutes, providing the software developer with a usable tool. The prototype can drive a target system using actual display and input devices, as well as other peripherals. One software engineer can run billions of validation cycles on the design before final silicon. If the engineer finds problems, the engineer can fix them in the RTL without the expense of developing software workarounds or sacrificing end-product performance. Even better, the software work is parallel, and the end product comes out sooner.

Aptix offers an integrated set of hardware and software products that allow engineers to partition complex designs across multiple FPGAs and integrate them with existing hardware to complete the design. Axis Systems, now merged with Verisity (www.verisity.com), takes advantage of reconfigurable-computing techniques to provide both emulation and accelerated simulation of hardware in its Xcite and Xtreme products.

Cadence's Quickturn group markets the Palladium family of products. Palladium products are fully compatible with Cadence's Incisive verification platform and provide both emulation and acceleration. Users can share one product within the network to perform simulations on the same emulated hardware. In emulation mode, they support full system verification by incorporating peripherals, embedded processors, and embedded software. EVE (Emulation and Verification Engineering) developed ZeBu to decrease the costs of traditional emulators and maintain the required high perfor-



Figure 1 The XT2000 allows engineers to prototype various processors architectures.

mance. The product can emulate designs with a complexity of 1 million to 12 million ASIC gates.

The CelaroPro, Mentor Graphics' latest entry in the emulation-product family, addresses the challenges of the industry's largest designs. Expanding the emulation function beyond pure chip emulation, CelaroPro creates a powerful hardware and software co-verification option with advanced functions, including code coverage, testbench acceleration, interactive design debugging, and emulation within C-based environments.

Companies offering processor cores that engineers embed in their designs often provide emulation systems, as well. ARM distributes the RealView Baseboard, which supports 3-D-graphics application development around ARM and Power/VR MBX cores. The system provides an AMBA multilayer prototyping environment and includes both extensive memory and a list of possible peripheral controllers. It supports expansion-module sites for both static and dynamic memory, logic-tile-expansion sites for peripherals and secondary processors, a PCI backplane, clock generation and arbitration, and a choice of LCD panels.

Tensilica offers the low-cost XT2000 development tool, which uses a programmable-logic device to emulate an Xtensa-processor configuration. The emulation kit enables the developer to evaluate various processor-configuration op-

tions and to develop and debug software early in the design cycle (Figure 1).

PLATFORM-BASED DESIGN

Professor Alberto Sangiovanni-Vincentelli, the Edgar L and Harold H Buttner chairman of electrical engineering at the University of California—Berkeley, coined the phrase “platform-based design” to describe a method of developing ICs using a nucleus of standard cores. The platform almost always comprises a microprocessor core and some peripheral-control devices that are inherent to the application area that the product is targeting. Example platform options include a graphics platform, a network-controller platform, and a wireless-communication platform. Taken to its logical endpoint, platform-based design is the implementation in silicon of the traditional method of developing software for a known hardware architecture. The only difference is that it offers the freedom to add product-specific logic on the same IC that contains the platform.

Two trends in the industry signal that, in the next few years, most ICs will be built using platform-based design. Semiconductor-fabrication technology has reached a temporary practical limit. The engineering complexities of developing an IC to be manufactured at either 130- or 90-nm-process nodes put the choice to design a traditional ASIC for these nodes out of the engineering and financial capabilities of most system houses. The next step, to 65 nm, presents even more engineering issues. Although some EDA tools address platform-based design, the method does not require any specific EDA tool, except models of the cores that make up the platform. The models are available as either bonded-out cores for use with emulation products or software models at various levels of abstraction—from transaction models that allow engineers to verify the proper implementa-

FOR MORE INFORMATION...

For more information on products such as those discussed in this article, contact any of the following manufacturers directly, and please let them know you read about their products in *EDN*.

AdvEDA
www.aveda.com

Aptix Corp
www.apnix.com

Cadence Design Systems
www.cadence.com

Emulation and Verification Engineering
www.eve-team.com

Spiratech
www.spiratech.com

Tenison EDA
www.tenison.com

VAST
www.vastsystems.com

Aldec
www.aldec.com

ARM
www.arm.com

CoWare
www.coware.com

Mentor Graphics
www.mentor.com

Summit Design
www.sd.com

Tensilica
www.tensilica.com

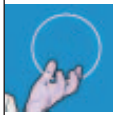
Xilinx
www.xilinx.com

Altera
www.altera.com

Axis Systems
www.axiscorp.com

Synopsys
www.synopsys.com

The Mathworks
www.mathworks.com



tion of bus protocols, to RTL models that you can use to ensure that the design meets timing, power, and signal-integrity requirements.

Designers using FPGAs have less freedom in place-and-route choices, but platforms generally restrict such freedom in ASIC design. More systems companies are opting to trade some freedom of choice that requires extensive analysis and knowledge of fabrication processes for shorter and cheaper design cycles. Altera offers both platforms built around its Nios RISC processor and an inventory of proven

cores from its AMPP (Altera Megafunction Partner Program) organization. Xilinx provides a number of platforms built around the IBM PowerPC processor as well as a number of proven cores from its AllianceCORE program. Both ARM and Tensilica offer platforms that you can use with a number of foundries when designing an ASIC. Cadence, Mentor, and Synopsys offer programs that allow their customers to use cores from certified providers, as well as design services to increase the effectiveness of a system-design team in completing an ASIC design.

A modified approach to platform-based design allows engineers to build virtual platforms that provide software developers with a model of the hardware. This approach blends ESL techniques with platform-based design. It allows more freedom in the platform architecture; thus, engineers can develop proprietary platforms. This approach, supported by products such as CoMet and Meteor from VAST (Vast Systems Technology), is feasible only if the system company wants to build a family of products around a proprietary platform. In all other instances, the resources and time spent developing a proprietary platform would increase the cost of the product and prevent it from becoming competitive in the market.

Summit Design also crosses the boundary between platform-based design and ESL. Its Visual Elite ESC sup-

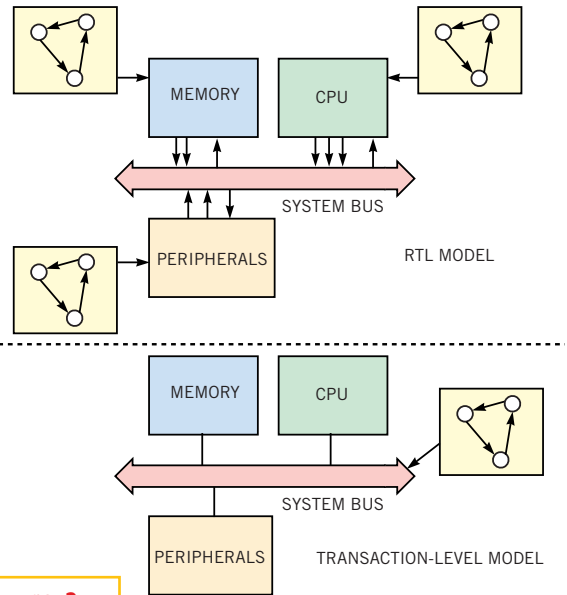


Figure 2

Transaction-level models are easier to develop and debug than corresponding RTL models.

ports SystemC modeling but offers an integrated ISS (instruction-set simulator) for the Xilinx PPC405 processor, the ARM CCM family of models, and Motorola's (www.motorola.com) MPC7410 and MPC7450. The tool links with the vendor-specific software-development-environment tools. It allows interfacing with the ISS at a signal level or a transactional level for higher execution speeds and provides fully synchronized hardware/software debugging and visibility.

Aldec has entered the embedded-system market with CoVer, a software-development environment and hardware-verification system that supports the Altera Nios and Xilinx MicroBlaze processors.

ESL TOOLS

After many years of ESL's existing as nothing more than a promise, it has finally established a market in the EDA industry. The tools are still somewhat immature, but they are making much progress. Unfortunately, the years of struggles have forced some entries, mostly start-up companies, to abandon efforts to develop tools in this area. The major problem facing EDA companies that rely only on this sector for revenue is that tool prices are lower than they should be. Users seem willing to pay a lot of money for tools that address the back end of the design flow, but they are unwilling to pay a lot for design-planning and -entry

tools, despite the value such tools can deliver. Good initial design can save many weeks later in the development cycle, but the savings are not immediately apparent.

Most of today's ESL tools focus on providing software engineers with an early model of the hardware to allow software development to proceed in parallel with hardware-design refinements. Synopsys was the original major proponent of SystemC, and its Discovery Verification Platform allows mixed-HDL (hardware-description-language) simulation and system-level verification by supporting SystemC models. Engineers can span the spectrum of modeling abstractions within the platform from transaction- to transistor-level modeling.

CoWare is an original ESL pioneer and has successfully weathered some start-up problems. It recently entered an engineering and business partnership with Cadence that should secure its existence. Its ConvergenSC uses the SystemC modeling environment. SystemC is more convenient for EDA vendors than for designers, but, while new and more appropriate options are under development, SystemC provides a way for design teams to evaluate the design at the architectural level and even simulate software and hardware models that approximate the behavior of digital logic. Users of ConvergenSC can now link their models to Cadence's Incisive platform and take advantage of the simulation environment's multilingual-modeling capability to refine the hardware models until they are precise representation of hardware behavior. ConvergenSC supports transaction-based modeling, a method that allows engineers to describe the communication portion of a system by modeling the messages sent on a system bus. It helps designers determine whether the functions are correctly distributed within the system by eliminating or minimizing bus contentions. Using transaction-based modeling simplifies the system model, making it easier to debug (Figure 2). Of course, completing the implementation requires you to also develop and test the RTL model of the system.

Spiratech markets the Cohesive family of products, which aims to provide designers with

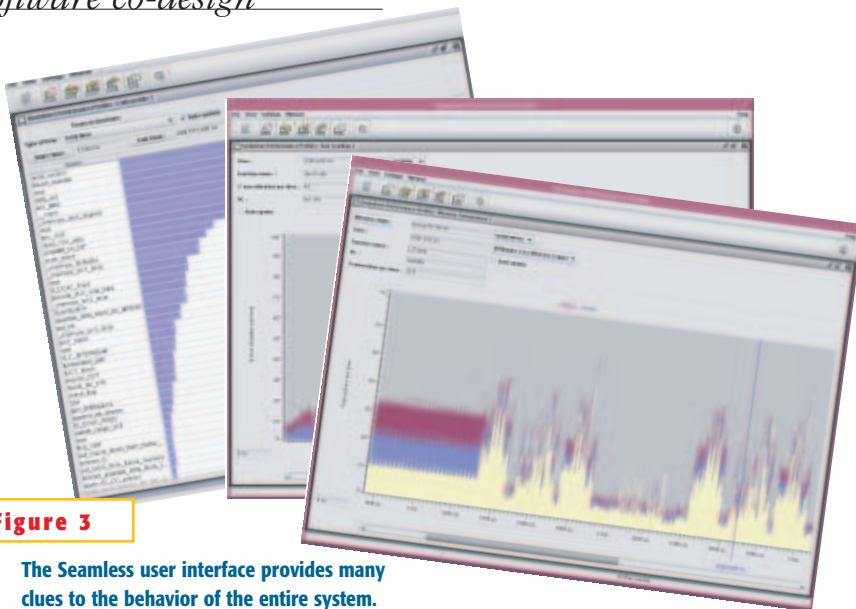


Figure 3

The Seamless user interface provides many clues to the behavior of the entire system.

a spectrum of views of the behavior of a design at various levels of abstractions. Its products are currently compatible with Mentor Graphics' ModelSim simulator and fully support designs that either use the PCI bus or communicate using UART circuitry. Another company, Tenison EDA, offers VTOC, which provides cycle-accurate models written in C++ or SystemC to support the early development of software drivers and applications. You can use VTOC with the Cadence Incisive platform. This strategy allows engineers to combine or replace the cycle-based models with more accurate RTL models written in VHDL or Verilog. A new company, AdvEDA, has just introduced the Miss Univers unified hardware-software co-verification option. It targets systems designed using multiple processors and uses one simulation kernel to control multiple ISS models and multiple RTL blocks. The goal is to reduce the effort required to develop testbenches. Testbench development consumes a significant amount of resources dedicated to testing and verification and is also prone to errors that compound the effort required to verify a design.

System architects have for years used The Mathworks' Matlab and Simulink to explore various algorithms in search of the best approach for their requirements. Recently, the two products have achieved much wider distribution and use, and a few traditional EDA vendors are considering integrating the products into their own ESL flow. The Mathworks, with as-

sistance from Mentor, has developed Link to ModelSim. As the name implies, it lets you cosimulate a design that is modeled partly in Simulink and partly in any modeling language that ModelSim supports. It significantly increases the viability of linking architectural and RTL design.

Mentor Graphics has for some time had a presence in the co-design market. Its product, Seamless, supports multi-processor designs and processors with multiple address spaces. A variety of commercial-processor models is available. Seamless uses the C-Bridge interface to support C models and is also integrated with a number of commercial logic simulators, such as ModelSim, NCSim from Cadence, and VCS from Synopsys. The integration allows engineers to use various levels of models, all written in common HDLs. A software engineer using Seamless can observe the behavior of the code through a number of windows. In particular, developers can see the code profile indicating which blocks are being executed (Figure 3). A bus-load profile shows how effectively the bus is being used, and a memory-transactions profile helps optimize the size of the cache versus the size of the central or local memory.

As work on new languages such as SystemVerilog or extensions of languages such as Verilog and VHDL progresses and designers gain experience in dealing with architectural-level issues, ESL tools will improve. □

You can reach Technical Editor Gabe Moretti at 1-941-497-9880, fax 1-941-497-9887, e-mail gmoretti@edn.com.



TALK TO US

Post comments via Talkback at the online version of this article at www.edn.com.