Everyone is talking about hardware/software codesign, but few companies are doing much about it. Knowing what EDA tools are available and how to apply them to systems on chips simplifies this difficult task.

Virtually all chips containing one or more processing engines need software to operate. You need to develop this software, whether on the chip in ROM or externally, with the chip’s hardware development. Unfortunately, the story about a project team’s hardware and software developers exchanging business cards when they meet approaches the truth. System-on-a-chip hardware and software design and implementation are often separate and parallel operations until the late stages of hardware development (Figure 1). This situation often produces specification and implementation mismatches between hardware and software components.

A design methodology that waits until a hardware prototype is available before checking software compatibility causes many problems. If you discover a software bug, you cannot easily change the hardware without a significant delay in final product release. Also, hardware description languages (HDLs), such as Verilog and VHDL, along with HDL simulators, have greatly accelerated chip-hardware development over the past few years. This acceleration puts an even heavier load on software-development teams to interface with hardware developers earlier in the system-design cycle. Fortunately, help is on the way in EDA tools and design methodologies that can ease your hardware/software codesign burden.

System partitioning
Many factors influence how you partition a system into hardware and software (Reference 1), including:

- Overall system performance, which determines how much of the design you must put into hardware to meet data-throughput specifications;
- Cost of system implementation, including development and physical cost of hardware and software development;
- Ability to modify the system software if it requires future changes, such as for an evolving communication standard;
- Computational nature, meaning how a system processes data;
- Hardware and software concurrency; and
- Communications-overhead cost between system hardware and software components.

Even though you cannot automatically do the partitioning, you can use tools from several software vendors, such as Alta Group, CPU Technology, and Synopsys (Table 1), to validate high-level system representations of partitionings and choose the best one. Today's tools, however, can only point you in the right direction. You still need to simulate your design at lower levels of abstraction to determine if you are meeting required speed, size, power, and cost specifications.

<table>
<thead>
<tr>
<th>Company</th>
<th>Product</th>
<th>Function</th>
<th>Price¹</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aisys Intelligent Systems</td>
<td>DriveWay</td>
<td>µC/µP drive-code generation</td>
<td>$195 to $1995</td>
<td></td>
</tr>
<tr>
<td>Alta Group</td>
<td>Envision</td>
<td>High-level system design/verification</td>
<td>$60,000</td>
<td>Focused on multimedia design</td>
</tr>
<tr>
<td>Aptix</td>
<td>System Explorer</td>
<td>Hardware emulation</td>
<td>$65,000</td>
<td></td>
</tr>
<tr>
<td>CAE Plus</td>
<td>ArchGen</td>
<td>Replicate boards</td>
<td>$12,000²</td>
<td></td>
</tr>
<tr>
<td>CPU Technology</td>
<td>SystemLab</td>
<td>Virtual-prototyping environment</td>
<td>$69,900</td>
<td>Design dependent</td>
</tr>
<tr>
<td>Eagle Design Automation</td>
<td>Eaglei</td>
<td>Hardware/software codevelopment</td>
<td>$40,000³</td>
<td>Virtual-prototyping service</td>
</tr>
<tr>
<td></td>
<td>Eaglev</td>
<td>ASIC validation tools</td>
<td>$40,000³</td>
<td></td>
</tr>
<tr>
<td>Ikos</td>
<td>VirtuaLogic SLI</td>
<td>Hardware emulation</td>
<td>$150,000⁵</td>
<td></td>
</tr>
<tr>
<td>I-Logix</td>
<td>Magnum</td>
<td>Hardware/software codevelopment</td>
<td>$25,000⁴</td>
<td></td>
</tr>
<tr>
<td>Mentor Graphics</td>
<td>Seamless CVE</td>
<td>Hardware/software cosimulation</td>
<td>$75,000</td>
<td></td>
</tr>
<tr>
<td>Quickturn Design Solutions</td>
<td>System Realizer</td>
<td>RTL emulation</td>
<td>$240,000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>HDL-ICE</td>
<td>RTL in-circuit-emulation software</td>
<td>$45,000</td>
<td></td>
</tr>
<tr>
<td>Synopsys</td>
<td>COSSAP</td>
<td>DSP-design tool suite</td>
<td>$40,000⁶</td>
<td>Code generation for DSP chips</td>
</tr>
<tr>
<td>Zycad</td>
<td>Paradigm RP</td>
<td>Rapid prototyping system</td>
<td>$30,000</td>
<td></td>
</tr>
</tbody>
</table>
### Table 1—Examples of EDA-tool and -equipment vendors for hardware/software codesign

1. Starting prices, unless otherwise noted.
2. Used with an existing System Explorer emulation system.
3. Price excludes Virtual Software Processor models; prices for those models start at $5000.
4. Price excludes Sharpshooter code generators; prices for those generators start at $10,000.
5. Also requires a logic analyzer for operation (not included in price).
6. Hardware- and software-implementation kits are available for $66,100 and $50,000, respectively. Both include COSSAP.

### Cosimulation techniques

Throughout your system design, you need to simulate the hardware and software to see whether they work together correctly. Cosimulation lets hardware simulators and hardware models interface with system software. This interface lets you concurrently verify the software while exploring hardware trade-offs. Each type of simulation involves trade-offs between simulation speed and accuracy. You need simulation environments that adequately handle the software and hardware synchronization. You also need to be able to debug the system when problems appear. Debugging involves being able to "see" inside the internal states and register contents of hardware modules. Another factor to consider is the availability of simulation models, both for hardware and software modules. Table 2 from Reference 2 compares cosimulation techniques.

<table>
<thead>
<tr>
<th>Method</th>
<th>Speed (instructions/sec)</th>
<th>Debug capability</th>
<th>Model complexity</th>
<th>Turnaround time</th>
<th>Software checking?</th>
<th>Hardware checking?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nanosecond accurate</td>
<td>1 to 100</td>
<td>Best</td>
<td>Hardest</td>
<td>Fast</td>
<td>OK</td>
<td>Yes</td>
</tr>
<tr>
<td>Cycle accurate</td>
<td>50 to 1000</td>
<td>Excellent</td>
<td>Hard</td>
<td>Fast</td>
<td>OK</td>
<td>Yes</td>
</tr>
<tr>
<td>Instruction level</td>
<td>2000 to 20,000</td>
<td>OK</td>
<td>Medium</td>
<td>Fast</td>
<td>Yes</td>
<td>OK</td>
</tr>
<tr>
<td>Synchronized handshake</td>
<td>Limited by hardware simulation</td>
<td>No processor states</td>
<td>None</td>
<td>Fast</td>
<td>Yes</td>
<td>OK</td>
</tr>
<tr>
<td>Virtual hardware</td>
<td>Fast</td>
<td>None</td>
<td>None</td>
<td>Fast</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Bus functional</td>
<td>Limited by hardware simulation</td>
<td>No processor states</td>
<td>Easier</td>
<td>Fast</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Hardware modeler</td>
<td>10 to 50</td>
<td>No processor states</td>
<td>Timing only</td>
<td>Fast</td>
<td>OK</td>
<td>Yes</td>
</tr>
<tr>
<td>Emulation</td>
<td>Fast</td>
<td>Limited</td>
<td>None</td>
<td>Slow</td>
<td>OK</td>
<td>OK</td>
</tr>
</tbody>
</table>

The type of model you use to represent a hardware component reflects a trade-off between the accuracy, or detail, of the part you are modeling and the speed of simulation. The most precise are nanosecond-accurate models, which have accurate timing information at the model's pins and include complete functionality. This type of model is good for checking system hardware details. The high accuracy comes at a price, however. The model is difficult to design, and its simulation is slow. You may also have problems getting a nanosecond-accurate model of a particular component.
Manufacturers of this part, fearing reverse-engineering of a model into a competitive part, may not make such models available. Of less complexity is the cycle-accurate, or zero-delay, model. This model supplies correct pin transitions at the edge of each clock transition. Fewer unique event times during simulation result in a simulation speed-up of a factor of approximately 10 over a nanosecond-accurate model. A still-faster model is one that is instruction-set-accurate. This model, also called an Instruction Set Simulation (ISS) model, correctly emulates the instruction set, meaning that the model correctly shows register and memory values but provides no timing information. This type of model is generally not useful for hardware evaluation but is good for debugging software. ISS models do not correctly model some complex processor functions, such as pipelining timing problems, but the model simulates faster than does a cycle-accurate model. Generally, the faster the model, the less accurately it models the system's hardware and the faster it simulates.

Several EDA companies offer tools for the hardware/software codesign problem; none of these tools is currently completely satisfactory. Moving a step toward "easier" hardware/software co-verification is Mentor Graphics with the recent announcement of the Seamless CoVerification Environment (CVE). Seamless CVE is the first product stemming from last October's merger of Mentor Graphics and Microtec Research (Santa Clara, CA), a software-development-tool maker. Seamless CVE lets software developers interact with hardware models at the behavioral level and register-transfer level (RTL). The tool's cosimulation kernel works with normally unrelated hardware- and software-design tools (Figure 2). Mentor claims that Seamless CVE performs simulation two to three times faster than does a gate-level simulation.

Seamless CVE presents a true processor model to a logic simulator. The model interacts with the system bus, so that the system correctly models and executes software and hardware interactions. The cosimulation kernel uses synchronization algorithms for timing, optimization algorithms to speed simulation, and memory-management logic to speed data transfer between hardware and software memory models. Seamless CVE supports several popular Verilog and VHDL simulators, including the company's own QuickSim II and QuickHDL and Cadence's (San Jose, CA) Verilog-XL. For very high-performance applications, you can use a hardware accelerator to speed simulation. Microtec's Xray debugger eases software development and source-code tracing. Xray links to commercial and proprietary real-time operating systems (RTOSs) and to compilers for C, C++, and assembly language. Mentor also offers process and DSP-engine models of Advanced RISC Machines (Los Gatos, CA), Hitachi (Brisbane, CA), Intel (Folsom, CA), Motorola (Austin, TX), SGS-Thomson (Lincoln, MA), and Texas Instruments (Dallas) to assist system development.

Virtual prototypes

You use a virtual prototype in place of hardware to help define your system and to debug it after you define the hardware. The idea of a virtual prototype, a software model of a hardware system or subsystem, is conceptually simple but difficult to implement. When you use these models early in the design process, they let you do what-if analyses to decide what portion of a system you should implement in hardware and software. Cost, size, power, and time-to-market constraints are your guides in making this decision. Simulating system software on virtual prototypes helps you make hardware trade-offs; for example, should you put a larger cache on an ASIC processor chip or use faster and more expensive memory chips? After you decide what the hardware should look like, using a virtual prototype to test the application or embedded software tells you if the choice of hardware implementation and software definition is correct. If the software functions incorrectly, you must decide whether to change it or to modify the proposed hardware. Both of these tasks are still relatively easy, because you have not yet implemented your system in real hardware.

You can define virtual prototypes at various stages of your hardware's development: the conceptual,
architectural, RTL, or even gate level. The earlier you define and simulate a virtual prototype, the easier it is to make any necessary changes if the system functions incorrectly. Another advantage of using high-level virtual prototypes is that you can simulate them faster than you can lower level software models. However, because you are using software models in place of real hardware, simulation is much slower than emulating the actual hardware or using a hardware prototype.

One company, CPU Technology, offers a virtual-prototyping service using the company's SystemLab environment. SystemLab is a virtual environment that uses fully functional, clock-accurate software models and representations of the equipment you would normally use to design and validate your hardware/software system. Such equipment includes an in-circuit emulator, a logic analyzer, and a clock generator (Figure 3). CPU Technology develops virtual prototypes of the hardware blocks, including off-the-shelf components and ASICs, of your system. You use these prototypes to prove system concept, evaluate hardware architectures, and integrate software and hardware. You can also use SystemLab to modify a system before physically implementing changes. The company's models are cycle-accurate at the instruction level and allow you to run reasonably fast simulations. For example, an 8051 processor simulates in SystemLab at about 8000 clock cycles/sec on a Pentium system and about 15,000 clock cycles/sec on a system with a Pentium Pro processor. You can also run complex system simulations in SystemLab, because you only need about 10 bytes of memory for each equivalent logic gate in a design. (A 1 million-gate ASIC has a modest requirement of about 10 Mbytes of memory beyond that needed for the operating system.)
RASSP program improves design process

Kevin Jorgensen, Precedence

In June 1993, the Advanced Research Projects Agency (ARPA) initiated a program to investigate and develop design methodologies to improve the design process for signal processors. ARPA chartered the Rapid Prototyping of Application Specific Signal Processors (RASSP) program to improve by a factor of four the time required to take a design from concept to prototype, design quality, and life-cycle cost. To develop technology for the RASSP program, ARPA selected EDA vendors Alta Group, Berkeley Design Technology (Fremont, CA), Mentor Graphics, Precedence (Santa Clara), and Quickturn Design Systems. The program's challenge was to leverage these vendors' expertise through a new unified design environment and to accelerate delivery of signal-processing hardware and software.

In a typical RASSP design-environment application, designers consider the trade-offs between implementing signal-processing algorithms in hardware using off-the-shelf processing cores and using software that loads into onboard ROM. RASSP divides the process into milestones, each representing an incremental refinement of the design and a step toward the virtual prototype and final implementation. Designers use commercial simulation technologies from RASSP participants to evaluate trade-offs at each milestone. A simulation backplane provides the technology that links the commercial point tools into a contiguous flow to facilitate an implementation path. Designers create a model at each milestone; they exercise this model using various simulator combinations. Partitioning is the creation of a model of the design. In an RASSP design process, hardware and software partitioning occurs early in the design process, as well as between various hardware alternatives, as the designer refines the final implementation. Partition-creation ease and the speed at which designers can evaluate partitions determine efficiency of the design process.

Although the RASSP program is specific to DSPs, the resulting developments apply to broad range of designs. As the first complete hardware/software-codesign environment based on commercial-tool technology, the RASSP program represents an important turning point in the development of complex design solutions. You can find additional RASSP information on the Web at http://esto.sysplan.com/ETC/RASSP/.

Eagle Design Automation's Eaglei addresses the embedded-system codesign market. Eaglei contains the Virtual Product Console (VPC) and the Virtual Software Processor (VSP). VSPs are software models of popular µPs to use with the VPC, which links the models with application software. There are two types of VSPs available. The first is an enriched bus-functional model, composed of a C-code core in a Verilog or VHDL wrapper. The C code communicates with the application software, and the HDL wrapper represents the system hardware for an HDL simulator. Eagle's recently introduced ISS models represent the second kind of VSP.

Using the VPC and a VSP with a hardware simulator provides a hardware/software codevelopment environment (Figure 4) that can simulate as many as 5000 instructions/sec. The VPC also lets you test software against the hardware model at the behavioral, register, and gate levels of abstraction. Eagle offers VSPs for the Advanced RISC Machines ARM7; AMD (Sunnyvale, CA) 29K and X86, Intel X86 and i960; Mips Technologies (Mountain View, CA) 4000, and Motorola 68K and Power PC. Eaglei hardware-simulator support includes Cadence Verilog-XL, Chronologic's (Los Altos, CA) VCS,
Model Technology's (Beaverton, OR) V-System/VHDL, Vantage's (Fremont, CA) Optium, and Viewlogic's (Marlborough, MA) Fusion.

**Hardware emulation**

Moving down the design "food chain," emulation is a way of checking hardware and software compatibility early in a design at the RTL before synthesizing, as a logic-gate representation, or at the gate level. At these stages, the hardware is the functional equivalent of a breadboard. The emulation system configures your design within an array of PLDs, which can be commercial FPGAs or, for Mentor Graphics' SimExpress, in a full custom chip. Emulation occurs as fast as tens of megahertz. However, the entry cost of complex design emulation can be hundreds of thousands of dollars for a system with a capacity of hundreds of thousands of gates. Also, the time required to set up a complex system for emulation may be weeks or months (see box, "Chip-emulation advantages"). However, actual emulations take only hours, and you can rerun them as necessary when you make software or hardware changes.

---

**Chip-emulation advantages**

Setting up a complex system for emulation can require a large time commitment. Hamid Butt, the director of system development of the Business Development Unit of Toshiba America, explains how the company uses emulation for embedded-processor and MPEG II chip designs. The chips his group designs typically have about 200,000 gates of logic plus memory. After RTL verification, Toshiba compiles the Verilog description and emulates the design on a Quickturn system. The setup time for the chip to this point in the design ranges from two to three months. The actual emulation time is a few hours (ranging from about 8 hours for an incremental compilation to about 18 hours for a full compilation). It typically takes just a few hours to make any change emulation points out in the software. A required hardware modification, at the RTL, may take one to seven days to implement.

A big advantage of using emulation at RTL rather than simulating the chip with a Verilog simulator is the speed of the emulation: a three-order-of-magnitude improvement. Toshiba ran a benchmark on an MPEG-II chip with Verilog simulation using Chronologic's VCS simulator running on a 167-MHz UltraSPARC workstation. The simulation took 3½ hours to simulate 1½ frames. At a 32-frame/sec rate; this speed translates to more than three days to simulate 1 sec of data. Using emulation, Toshiba emulated the same chip in about 5 sec, an improvement by a factor of almost 1700.

Besides Mentor Graphics, companies such as Aptix, Ikos, and Quickturn Design Solutions offer emulation hardware and software. Of this group, Quickturn is the best known in the simulation arena for the company's System Realizer product line. Quickturn performs RTL emulation before synthesis. The company's top-level system can emulate systems with as many as 3 million gates as fast as 4 MHz. The company's new Quest II emulation software reduces the time to compile a design before emulation and enhances the visibility of the design during debugging. Toshiba America (San Jose, CA) uses a Quickturn emulation system to verify chip hardware at the RTL and gate level (Figure 5). Toshiba performs an optional functional check after synthesis to the gate level if a custom-designed block is on the chip. An example of such a block would be the datapath for a processor, because the custom design is more area-efficient than one done with automatic EDA tools. Toshiba does timing simulation and verification in parallel with chip floorplanning and layout to save design time. Both subflows are closely coupled; a problem in either layout or timing usually means a change.
Looking Ahead

Cosimulation techniques with virtual prototypes that replace actual hardware are becoming more common. Simulation needs to accelerate, however. Even the 20,000-instruction/sec speed of Instruction Set Simulation (ISS) models is slower than the real-time operation of the final systems. Look for the development of different hardware models that allow you to simulate at higher speeds, albeit with a loss of model detail, when you first simulate the system for "proof of concept."

Another area needing assistance is system partitioning into hardware and software components before implementing hardware and code. EDA tools, such as Synopsys' COSSAP, let you graphically build a DSP system using predesigned and user-specified blocks. You simulate the system at a very high level of abstraction and explore algorithmic, architectural, and implementation trade-offs. After optimizing the system, you manually partition the blocks for either hardware or software implementation. COSSAP then helps you generate C code to run on industry-standard DSP cores or chips and either VHDL or Verilog for the hardware modules. Future system-design tools from EDA vendors will include ASIC capability in the design flow and semiautomatic partitioning of the system blocks into software and hardware modules. Fully automatic partitioning is difficult and will not appear in commercial tools for at least the next three to five years.

Hardware prototypes

The most technically ideal way to test your application software on system hardware is to build and use a prototype of the hardware. Software runs at or near full system speed on a prototype and provides accurate hardware-timing checking. Using the real thing gives you a range of software-development tools from which to choose when testing and debugging the hard-ware/software combination. However, if you find a hardware problem, it will most likely effect system-design time; a problem that necessitates a redesign might force the final product to miss a critical design window. This situation emphasizes the necessity of doing everything possible to run hard-ware/software coverification earlier in the design cycle.

The EDA industry, through new tools and design methodologies, is attacking the bottleneck of hardware/software codesign for complex chips (Table 2). Current efforts focus on the use of virtual prototypes for as-yet-unimplemented hardware and speeding the cosimulation of hardware and application software early in system design. The problem of initial system partitioning into hardware and software components, based on such constraints as data throughput, cost, and power, is still in evidence. Such partitioning is largely a manual operation based on designer expertise, and EDA and hardware vendors still need to develop techniques to further automate this procedure.

You can reach Technical Editor Jim Lipman at (510) 606-1370, fax (510) 606-1563, email ednlipman@mcmail.com

References


<table>
<thead>
<tr>
<th>Representative manufacturers of EDA tools and equipment for hardware/software codesign</th>
</tr>
</thead>
</table>
| **Aisys Intelligent Systems**  
Petach-Tikva, Israel  
(800) 397-7922  
fax (800) 625-5525  
www.aisys-usa.com | **Alta Group**  
Sunnyvale, CA  
(408) 733-1595  
fax (408) 523-4601  
www.altagroup.com | **Aptix**  
San Jose, CA  
(408) 428-6200  
fax (408) 944-0646  
www.aptix.com |
| **CAE Plus**  
Austin, TX  
(512) 338-0165  
fax (512) 338-1092  
www.cae-plus.com | **CPU Technology**  
Pleasanton, CA  
(510) 224-9920  
fax (510) 227-0539 | **Eagle Design Automation**  
Beaverton, OR  
(503) 520-2300  
fax (503) 520-2323  
www.eagledes.com |
| **Ikos**  
Cupertino, CA  
(408) 255-4567  
fax (408) 366-8699  
www.ikos.com | **I-Logix**  
Andover, MA  
(508) 682-2100  
fax (508) 682-5995 | **Mentor Graphics**  
San Jose, CA  
(408) 685-7000  
fax (408) 685-1202  
www.mentorg.com |
| **Quickturn Design Solutions**  
Mountain View, CA  
(415) 967-3300  
fax (415) 967-3189  
www.quickturn.com | **Synopsys**  
San Jose, CA  
(415) 962-5000  
fax (415) 965-8637  
www.synopsys.com | **Zycad**  
Fremont, CA  
(510) 623-4400  
fax (510) 623-4550  
www.zycad.com |

Copyright © 1996 EDN Magazine. EDN is a registered trademark of Reed Properties Inc, used under license.