Over the recent past, the focus of EDA tools has been on tackling the chip issues. Today, system design issues are emerging as the top concerns of design teams. The issues include the inability to simulate at the system level, incomplete and inaccurate system specs, and immature hardware/software co-design.

Co-design is an essential element of new design methodologies for system development:
- ratios of seven software engineers per hardware engineer are now common in the development of complex embedded systems;
- errors cost 10 times more to correct at the integration stage than during design;
- system specifications change impacting both hardware and software development;
- interfaces between hardware and software are non-trivial for system design.

A comprehensive hardware/software co-design methodology needs to deliver a unified implementation environment addressing the issues outlined above. What kind of tools can meet this challenge? Are EDA vendors close to providing the tools that the designers need?

**Brian Bailey, Mentor Graphics, Wilsonville, OR**

Over the past few years a number of commercial products have been introduced that address various parts of the system verification problem. The recent introductions have focused on the hardware/software boundary. These tools are a start, but the field is very much in its infancy. Today's tools are limited to a single abstraction of software and in most cases a single hardware simulation session. This makes the tools useful as an implementation verification solution and enables the notion of a virtual hardware prototype upon which we can perform software verification. It does not make them useful for higher level design or concept validation. This requires tools that are much more flexible and can cross many abstraction domains. At the same time, these hardware/software simulators must join with other parts of the system verification problem to create a complete verification solution. As with all simulation solutions, models are a key issue, both in terms of availability and accuracy. It appears as if many core providers are stepping up to the challenge of providing these simulation models for processors or are willing to assist in the validation of models. This will be a key step in the wide-spread acceptance of these tools.

**Amr Mohsen, Aptix, San Jose, CA**

The starting point for hardware/software co-verification is the availability of a high performance model of the hardware that enables running of the code. Performance is key to insure the basic code coverage. The model also needs to be highly flexible since hardware/software co-verification requires the ability to make design trade-offs - such as partitioning - to meet the objectives of the new system. The trade-offs are based mostly on meeting cost and performance objectives.

Another important consideration that will dominate the development of embedded systems in the future is co-verification of hardware and software in the system's final environment. A key variable in verification is the impact of the environment - typically unpredictable - on the overall system operating performance. Applications based on subjective datatypes (audio, video) will demand real-time co-verification platforms provided by prototyping to meet this challenge.

**Kurt Keutzer, Synopsys, Mountain View, CA**

There is no single hardware-software co-simulation problem and as a result there is no single tool or technology that can successfully solve all the problems associated with co-verification of hardware and software. Each developer must trade off the desired performance against the level of accuracy. Accuracy itself has two aspects. The first is the timing accuracy. The second might be called veracity, or the extent to which the model accurately models the design under verification. Once these parameters are fixed then the remaining challenge is to determine which solution comes closest to meeting the requirements at a cost, in time and money, that one is willing to pay. For hardware designers the trade-off is principally between software simulation and the relatively faster technologies of hardware acceleration, emulation or rapid prototyping. Not surprisingly, the faster technologies come at a higher cost in dollars and in development time. For software developers of embedded system application software the primary need is simulation speed and the trade-off is among a variety of simulation approaches which achieve greater degrees of speed at the cost of diminishing the accuracy of the modeling of the hardware. Tool vendors can aid design teams more by helping them to adequately determine their tool needs than by claiming unfounded capabilities for their tools that ultimately only serve to confound the customer.

**Richard Moseley, Motorola, Austin, TX**

Over the last several years, large design houses have successfully proved the value of hardware emulators through the verification of very large circuits. Using this technique, billions of instructions can be run on a circuit and compared against a standard to guarantee a working solution. With a new market for hardware / software co-design and verification, many vendors are claiming to have solutions available to provide for this need. The question is, for a typical design engineering team with a limited capital budget, is hardware emulation a viable solution to a hardware / software co-design effort? In this new role as a systems engineering co-design tool, will hardware emulation tools of today, which have proved so successful for large corporations with large designs, provide a solution for next generation design needs? With the present tool capabilities and cost, are the hardware emulator manufacturers able to satisfy the system designer with affordable tools that indeed decrease design time and produce superior designs with higher quality and lower risk?
Jim Rowson, Alta Group of Cadence, Sunnyvale, CA

Put the DESIGN into Co-DESIGN - The discipline of hardware/software co-design has, for most people, the connotation of relatively low level co-simulation of the software running on the hardware. Sometimes this happens using a processor model, and sometimes the software is run natively on the same machine as the hardware simulator. Co-design is much more than that. At the system level, many important trade-offs need to be made, not the least of which is the partitioning between hardware and software. At this architectural level the decisions are based on a combination of performance, cost, power, and business criteria. There are very few tools to help with this problem, Excel spreadsheets are the tool of choice. Given a top level architecture, there is then little help with synthesizing the software, hardware, and interface. Miscommunications between hardware and software teams about the interface cause problems when the two are integrated late in the design phase. A good synthesis step could remove this class of errors. Finally, after most of the work is done, we have to do a final diagnostic check to see that the software and hardware integrate correctly before going to prototype. This should be a last and mostly unsurprising step.

Geoff Bunza, Viewlogic/Eagle Design, Beaverton, OR

Hardware/software co-verification does not just occur at the system integration point but rather throughout the design process including in the ASIC development team at ASIC development time before fabrication, at the module level, at the board level, at the sub-system and at the system level. Co-verification needs to occur before fabrication for many reasons, including economic reasons and time to market motivation but also especially to facilitate communication between hardware and software design groups throughout the design process.

Willis Hendley, Sun Microsystems, Chelmsford, MA

No position statement available.