Hardware/Software integration: Closing the gap

Tom Huang, Pete Mar, InPA Systems inc. - March 13, 2011

Introduction
In recent years, system integration associated with System on Chip (SoC) design has grown at a rapid rate and continues to drive the semiconductor design market. Although this growth has been beneficial for the design community, the sophisticated and complex manufacturing requirements of next generation devices have increased the cost of ASIC and ASSP development. Product implementation of complex, low-power designs requires early integration of various hardware features with corresponding firmware onto one silicon device. The reduced lifespan of current products have condensed SoC development cycles and have made the SoC verification and in-system validation process an arduous task. The bulk of time spent during the product development cycle of SoCs is often in hardware/software integration and has brought in-system validation to the forefront of this extensive process.

Hardware/Software development environments
Virtual platforms provide application software developers a head start towards interfacing with the IP and writing essential algorithms. Firmware can be written to interface with the fabric logic and enable the architecture, instruction set, memory, interrupt and important aspects of the design can be solidified prior to freezing the RTL. However, abstract models force developers to maintain separate versions of the firmware code to support both the virtual and hardware platforms. Although interface-modeling software is often employed, early access to hardware provides deterministic results to ensure functionality and verify the development is aligned.

In parallel with early software development, RTL logic functionality is defined, and then verified through logic simulation. This method becomes more tedious and difficult later in the design cycle as the design grows and the entire system needs to be verified. Methods such as RTL simulation and emulation are common practices, however, their advantages are offset by the cost of speed. Logic simulation provides high visibility into important nodes but run magnitudes slower than a design’s real-time speed. Emulation is faster than simulation and the high cost of such systems may be prohibitive.

The verification design gap
Software development tools and debuggers execute code rapidly, from setting breakpoints to stepping through instruction code with no delay. Although these debuggers execute in real-time or near real-time, software tools cannot provide visibility into the hardware other than some registers and memories. Verilog and VHDL simulators provide nearly full-node visibility into the hardware on a cycle-by-cycle basis via waveforms for debugging. However, compared to software development tools, HDL simulation/execution time is typically four to five orders of magnitude slower than virtual platform. HDL simulation can only generate data a few thousand cycles per second, which is too slow to enable debug of the firmware.
Reducing the SoC development gap

As the delay in development time effects are more prominent amongst software developers, current tool offerings allow software developers to integrate hardware models with all of the interfaces into virtual platform. Hardware developers can implement various methods to make a hardware model available for software testing. Fortunately, FPGA vendors have steadily developed robust features, which have a vital role in making FPGA-based SoC prototypes a mainstream vehicle to aid designers with hardware verification and software development. FPGAs have increased in complexity with key features critical for prototyping such as increased amounts of memory and IP as well as system interfaces to DDRAM, PCIe, Ethernet and high-speed serial interfaces of over 10 Gb/s. Capacities of the largest FPGAs have grown to a point where SoC prototypes may even be created in a multi-FPGAs. Real-time speeds of an FPGA prototype and capacities are amongst the main reasons FPGA-based ASIC prototypes have thrived and continue to grow as a methodology.

Detection of software bugs can take over 50 percent of the overall software development time since developers must not only locate the problem in the code, but also resolve the issue without introducing other errors. With early access to an FPGA prototype, firmware developers can implement runtime and automatic debuggers to detect memory corruption and other runtime errors. Firmware development during the bring-up phase of an FPGA prototype and continued debug access throughout the verification process is critical.

Logic simulation has long been a method of RTL design verification due to cost efficiency and great signal depth. However, this methodology is magnitudes slower than the design’s clock speed. Access to an FPGA-based SoC prototype provides a platform that can run very fast, however, due to pin limitations and available local memory, visibility in FPGA-based prototypes is restricting. Vendor tools which implement the design into hardware is also limiting, as any RTL change requires P&R to be re-executed, which consumes numerous hours to run.

A combination of the benefits of logic simulators’ high-visibility and the real-time speeds of a prototype greatly enhance the software/hardware integration process. The InPA Full Visibility Technology offers full signal visibility during the vector simulation and in-circuit operations. Vector simulation captures all of the input vectors from the simulation test bench and reads the output vectors from the FPGA prototype. A snapshot of all design registers and memory in the FPGAs is taken and used by the InPA Technology to initialize vector simulation. This vital information is stored
in a vector database, which enables the InPA Technology to reconstruct all the signals, thus, providing the same visibility as an RTL simulator, magnitudes faster. This same technique is used in the in-circuit operation (where the stimulus is not the test-bench but actual interfaces) with the added vectors that are being captured at full FPGA speed. Faulty vectors in the FPGAs are captured and stored in the InPA Technology vector database along with the other signals of the design. Similar to the vector simulation mode, all signals are visible.

**FPGA prototyping challenges**

Although FPGA-based prototyping has become a standard SoC validation platform, limiting factors make system faults difficult to be identified and resolved. For example, detailed information must be collected to resolve system faults from bugs found in the driver. Flaws in the system architecture and hardware modifications to resolve these issues emphasize the need for both teams to design from the top down. Catching system faults early during the “bring-up” phase, gives hardware designers time to modify the RTL and verify the design simultaneously, while the design is still being defined.

The availability of hardware provides a bit exact, clock cycle accurate development system. Equally important, designers gain early opportunities for development of the diagnostic software to verify that the prototype board and components are working properly and that the RTL design is properly implemented in the prototype.

Any modifications to the system architecture late in the development cycle can cause major scheduling delays in a process that can already be long and tedious. Therefore, early evaluation of the system architecture for key components such as sizes of the FIFOs, buffers, bus bandwidths and amount of code space required can help minimize the amount of throwaway code and is significant to reducing verification time. The uncertainty of the root cause to be a firmware or hardware failure when a system bug is discovered can consume much of the development time, often multiple days and even weeks. The debug process is inherently iterative which requires the bug to be identified, reproducible and reduced to a manageable state for resolution. Software engineers can control the flow by reading registers and memory map information of the design to analyze the hardware behavior but require the collaboration with system and hardware engineers to determine the cause of the problem. Such issues may occur due to a misunderstanding of the specification, implementing the wrong control sequence, or the design may contain a true hardware bug.

The entire process of typical hardware/software integration may take a few months to debug once the system contains stable hardware for application software development and a demo of the system. Although the implementation of a single test environment for hardware and software verification has made strong advances, the individual debug solutions of in-circuit emulators (ICE) for software and embedded logic analyzers (ELA) for hardware, respectively, have yet to make similar advances. Current technologies have limited control and methodologies are continuously being developed to synchronize the two debug environments.

**Areas of debug concentration**

- **Bring up**
  - Verify prototype board and components are working properly
  - Verify RTL design is properly implemented in prototype

A co-simulation environment integrating industry-standard logic simulators such as Incisive®, ModelSim® and VCS® with the FPGA prototype is an efficient method of implementing the existing simulation and hardware environments. An application such as a smart-phone prototype greatly benefits from such a verification methodology. For example, the actual design may take less than 20 seconds to boot-up. However, an FPGA-based implementation may take up to three minutes to
execute at 30 MHz (ARM soft core) \( *3 \times 60 = 5400 \text{ M cycles} \). Logic simulations provide useful data at a block-level and for top-level connections because the boot-up process is nearly impossible to simulate and may take months or longer. RTL simulations are standard practice to verify functionality, but lack of hardware access produces an iterative debug flow that is time-consuming until meaningful data is captured. Co-simulation implements the existing test-bench used during logic simulation to stimulate the hardware on the FPGA prototype and verify that the FPGA outputs correlate to the RTL description.

The co-simulation capability is extremely valuable because real hardware is the DUT and can be enhanced if the simulation process itself could be faster. The RTL description of an FPGA prototype is constantly evolving early on and every ECO change requires another implementation of P&R. Leveraging the existing co-simulation environment, the InPA Technology captures the test vectors through PLI routines during co-simulation to accelerate the regression test. By storing the test vectors, the InPA Technology detaches from the existing logic simulator and uses the stimulus for vector simulation. Long simulation times are greatly reduced with vector simulation by nearly 100 times, enabling the ability to maneuver through the simulation space faster. During logic simulation, it is inherent to the designer to specify which signals are vital to monitor, however, this produces a limited scope of data that could be avoided if all or most signals were visible with efficient data file size. The InPA auto-probe technology enables full visibility into all nodes of the design for rapid debug analysis, without having to generate large simulation dump files.

**System-debug**
- Detect problematic area with software debugger and register/memory map probing
- Determine if the source of the bug is either in software or hardware
- Fix software problem using software debug process
- Issues related to the RTL must be resolved by the debug engineer

Once a prototype is powered up, booting issues such as driver integration and system architecture flaws may be encountered. Since the booting process can’t be simulated, a hybrid function of in-circuit emulators and embedded logic analyzers through software with the available hardware can give insight into the source of high-level system bugs. Embedding soft IP into the prototype design using the InPA Technology provides logic simulation-like visibility, at real-time speeds to track a booting sequence. Connecting a sophisticated internal logic analyzer, an Embedded Micro Machine (EMM), to the peripheral bus, can easily monitor any peripheral related issues. An EMM has unique programming capabilities via a proprietary language to monitor a subsystem that is connected to a processor in such a case. These EMMs provide access to important areas of the subsystem such as an IP’s unique address (range) and data (pattern). If any of the peripherals have problems, then an EMM can be connected to the peripheral bus and different scenarios programmed on the fly without re-implementing P&R to track the booting sequence until the problem block is found. Instantiation of these EMMs occurs in either RTL or gate-level netlist as black-boxes for efficient reuse in the design. In a simulation environment, full-signal visibility aids in the debug process. This in-circuit mode allows for EMMs to be cross-triggered amongst multiple FPGAs and boards, providing full visibility to accelerate system debug and the bring-up process.

**RTL-debug**
- Inherit system level instrumentation from software engineer
- Refine hardware debug search
- Capture faulty hardware scenario in simulation environment

**ECO RTL Technology solutions**
A solution that synchronizes control of the hardware and software debug environments while improving the visibility of system-level architecture facilitates efficiency and tighter integration.
Although current software simulation provides full design visibility, the snapshot of captured data is clock cycle-limited. Verification of a complex SoC can take hundreds of thousands or millions of clock cycles that consumes a large amount of the verification time. A prototyping methodology and sophisticated EDA technology is necessary to meet the current challenges from the initial diagnostic testing of bring-up and hardware/software system-debug, to the granular RTL-debug:

. **Bring-up**  
- Integrate today’s RTL testbench and simulation environments with the FPGA prototype  
- Verify correct RTL mapping and working hardware by applying the existing RTL testbench integrity test on the FPGA prototype  
- Bring the faulty mapping scenarios into an accelerated functional debugger to correct the faults

. **System-debug**  
- Control and synchronize the existing software debugger (ICE), hardware and embedded instrument (ELA) at run time to determine the source as software or hardware problems  
- Handle system level active debug techniques in a multiple FPGA environment

. **RTL-debug**  
- Capture faulty hardware scenario and debug in an accelerated functional debugger (simulation-like) environment, cross linking the RTL and FPGA for debug ease  
- Full visibility within the faulty block to significantly reduce FPGA P&R iterations

**Conclusion**  
As a verification methodology, FPGA-based ASIC prototyping has brought the hardware and software debug environments onto a single platform. However, methodologies must evolve to reduce bring-up, system debugging, and RTL debugging time as primary goals. The current software and hardware debug tools offer advantages to the debug process in the two environments that can help reduce development cycles. By integrating an RTL simulation environment with the FPGA prototype hardware, a co-simulation flow can be implemented to automate design bring-up process. The InPA Full Visibility Technology aids in early software development by providing key insight into detailed nodes via hardware access. Implementation of diagnostics and verification of the system architecture with the FPGA prototype verifies essential functionality and provides vital control and synchronization of the software and hardware debug environments. This type of access to the entire prototype system creates a true system-debug (SoC development) environment for today’s complex SoC design projects.

Once system bugs have been identified in the FPGA prototype, the use of the InPA Technology soft IP enables advanced Active Debug™ capabilities to give full visibility and sophisticated control during validation. The InPA Technology can be implemented to interact with industry-standard simulators or directly with the prototype as it is stimulated by verified software peripherals. Debug analysis with a prototype running at real-time speeds and simulator-like visibility without reiterating through implementation greatly reduces the overall SoC development time.

InPA Systems is developing these patent-pending technologies to streamline the SoC bring up and debug process which will significantly compress the HW/SW integration gap when using FPGA prototypes, as illustrated above. We will follow this design article with a more detailed description of the InPA System product upon its release very soon.

**About the authors:**
Tom Huang, CTO/Founder, InPA Systems Inc.

Tom has founded or co-founded 6 companies and holds 9 patents in the last 20 years with over 33 years experience in semiconductor, system, ATE and EDA industries.

Before founding InPA Systems Tom’s EDA background includes roles as president and founder of BJ International LLC, CEO and founder of S2C, EVP and CTO of Aptix, CEO and founder of Osprey Design Systems, and co-founder and CTO of PIE Design.

Tom’s semiconductor background includes roles as, analog tester and design engineer at Micro Power Systems and National Semiconductor. Tom’s system background includes roles as test equipment development manager at Storage Technology and as a logic analyzer at Gould Biomation. Tom’s ATE background includes roles as CEO and founder of Gillytron and as product manager at Applied Optical Technology.

- **Pete Mar**, Sr. Field Applications Engineer

Pete Mar is a field applications engineer at InPA Systems. He received his BS degree in electrical engineering from Northeastern University. Mar’s industry experience includes consulting work focused on SoC validation with FPGA prototypes. He also has extensive EDA experience having supported ASIC and FPGA synthesis and verification tools during his time at Synplicity Inc.

For more information see: [www.inpasystems.com](http://www.inpasystems.com)