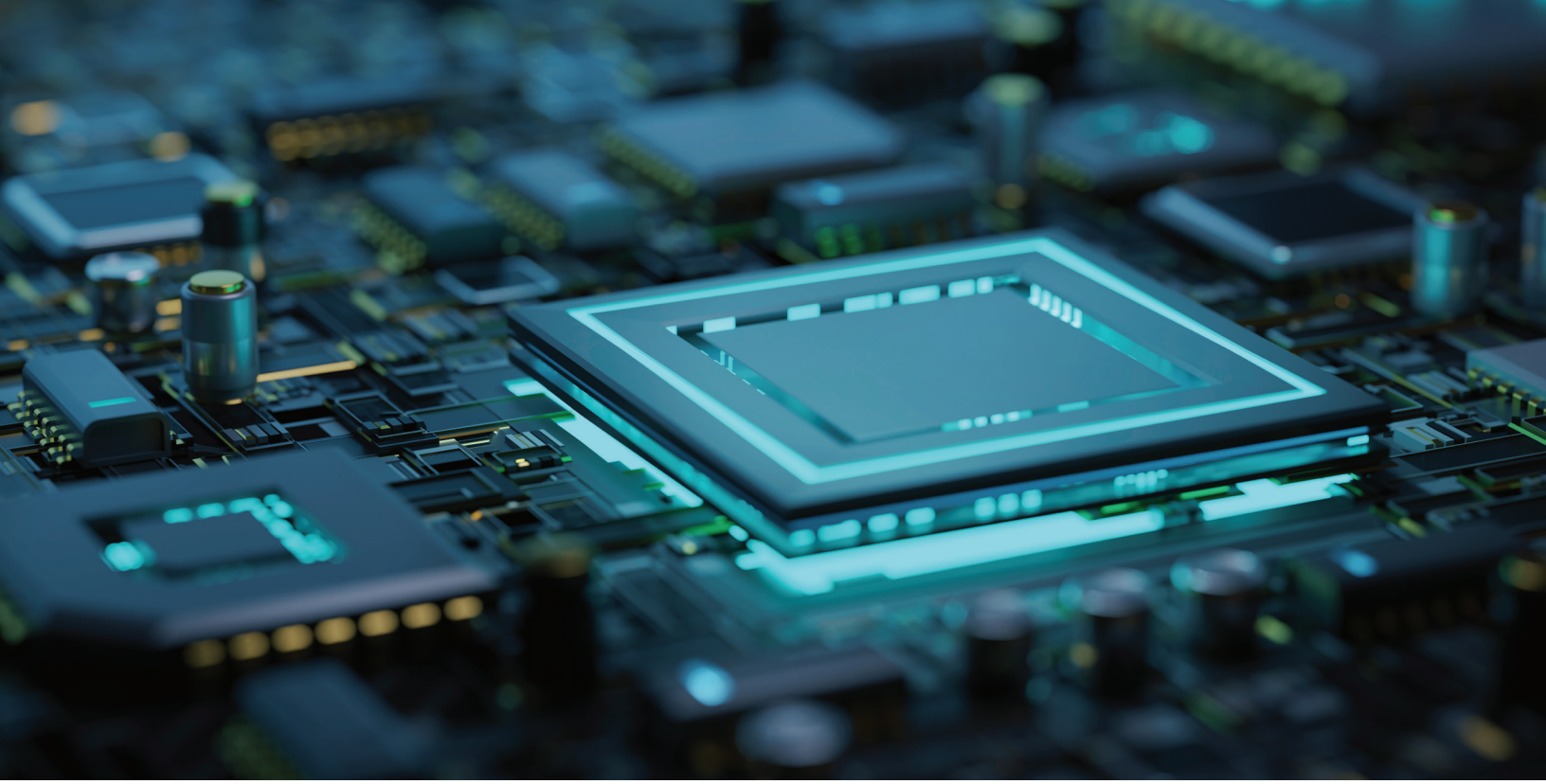


DIGITAL INDUSTRIES SOFTWARE

Veloce proFPGA supports firmware/ software development with a flexible FPGA prototyping system

Software complexity and what is necessary to achieve SoC verification

Executive summary
During a system-on-chip (SoC) design, the hardware team uses several different proxies to represent the circuitry at different stages of development. Early on, they rely on simulation with its near-total visibility and control for unit development. Later, many teams move the register transfer level (RTL) to a hardware emulation platform for intellectual property (IP) integration and functional testing. The most advanced emulation systems, such as the Siemens Veloce™ hardware-assisted verification platform, can support powerful system analysis tools at this stage as well.

But what about the needs of the software team?

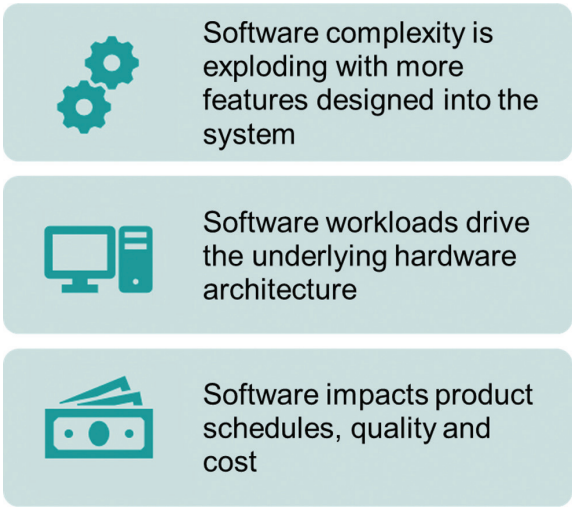
Gabriele Pulini

SIEMENS

Introduction: Software complexity and what it means to SoC verification

When first silicon arrives, the software team should be ready with a stable driver, operating system (OS) and application code–but developed on what? Application-level code can be developed on general-purpose systems or in the cloud, using stubs to represent the SoC and its peripherals. Software unit testing–up to the level of booting the OS–can be done conveniently on a hardware emulation system.

But then there are the tricky parts: integrating the hardware with the software or seeking root cause on obscure or seemingly nonrecurring bugs. What about the daily grind of software and IP regression testing? These tasks require an accurate model of the hardware and much of the visibility and controllability offered by emulation platforms. But they also need greater speed than emulation systems provide–as close to real time as possible. And by late in the design cycle, these tasks may be getting done at multiple sites around the world. It is important for the platform that supports them to be scalable and portable.



The answer to these requirements is often an FPGA-based prototype. Such a prototype can provide both hardware and software teams with a functionally correct implementation of the RTL and can run at or near target SoC speed. Ideally, the prototype can operate in-circuit mode and can still provide the needed visibility and control of internal signals.

FPGA challenges

Along with its benefits, field-programmable gate array (FPGA) prototyping brings some serious challenges. First, FPGAs are not trivial to use. Mapping an ASIC RTL design into an FPGA and achieving high operating speed, using the FPGA vendor's tools can require a deep understanding of the internal structures of the FPGA chips and the mapping algorithms. This internal structure includes not just the logic fabric, but specialized blocks such as Arm central processing unit (CPU) cores, caches, multiply-accumulators and memory instances, the use of which can greatly improve prototype performance.

Other issues can complicate the mapping as well. Clock structures on FPGAs can be quite different from those normally generated for SoCs. And if the RTL requires more than one FPGA chip–which will usually be the case for even modest-sized SoC designs–partitioning the RTL, distributing it across the multiple chips and linking the chips together can be an art form of its own.

Doing debug

Apart from just implementing the RTL, there are practical issues with FPGA prototypes. Designers must generally make their own provisions for accessing signals they want to observe, and for implementing trace buffers, trigger comparators and provisions to manipulate the clocks for single-step operation. These added signals must be routed out of the prototype to a host and comprehended in a debug user interface (UI). Ideally, this user interface would be consistent with that for other tools the design team is using. But more often than not, it is unique to the prototype.

With this added complexity, the prototype team often chooses to configure the prototype for a particular debug task and then reconfigure it, requiring an edit of the RTL, a recompile and possibly repeat steps to optimize performance–each time the debug task or the RTL changes significantly.

Veloce FPGA prototyping addresses software development

One result of these challenges is that well-funded SoC teams are likely to have a dedicated FPGA prototyping group, with its own FPGA specialists. Smaller teams may only build limited prototypes to explore particular portions of the design IP, or they may forgo FPGA prototyping altogether and just hope that code developed on surrogate systems works on the silicon. But it would be more efficient for large organizations, and more realistic for smaller ones, if an independent EDA vendor would deal with the challenges and provide an integrated hardware/software prototyping platform.

Recognizing the importance of FPGA prototypes to both the hardware and software sides of SoC design teams, Siemens EDA delivers a complete hardware/software prototyping platform, Veloce proFPGA platform. The platform, which is part of the Siemens Xcelerator business platform of software, hardware and services, directly addresses the challenges users have faced with FPGA prototypes (figure 1).

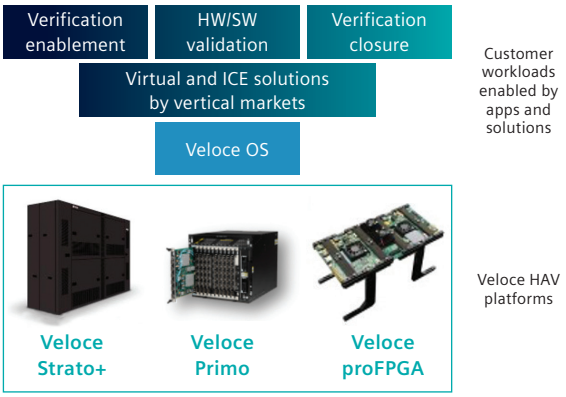


Figure 1. Veloce hardware-assisted verification family.

The Veloce proFPGA platform includes a range of enclosures and interchangeable FPGA modules, allowing it to scale from a single FPGA to 28 FPGAs and more without changes to the environment. The platform supports emulation of peripherals through RTL in the FPGAs and via at-speed in-circuit interfaces. It can automatically infer SoC memory instances using the internal configurable SRAM blocks in the FPGAs, or it can emulate SoC memory with external DDR4 boards.

Equally important to the power of the platform are the software tools, beginning with Veloce Prototyping Software (VPS): a complete set of tools that map the user's ASIC/SoC RTL design into the FPGAs, either fully automatically or under different levels of user's guidance (figure 2). VPS can infer opportunities to use special FPGA resources such as the static random access memory (SRAM) blocks. It can map a wide variety of SoC external memory interfaces onto a DDR4 interface on the prototype, allowing emulation of various external memory devices. VPS is also particularly adept at handling clock circuits, using FPGA clock structures and the logic fabric, recognizing and correctly mapping such tricky structures as gated and generated clocks.

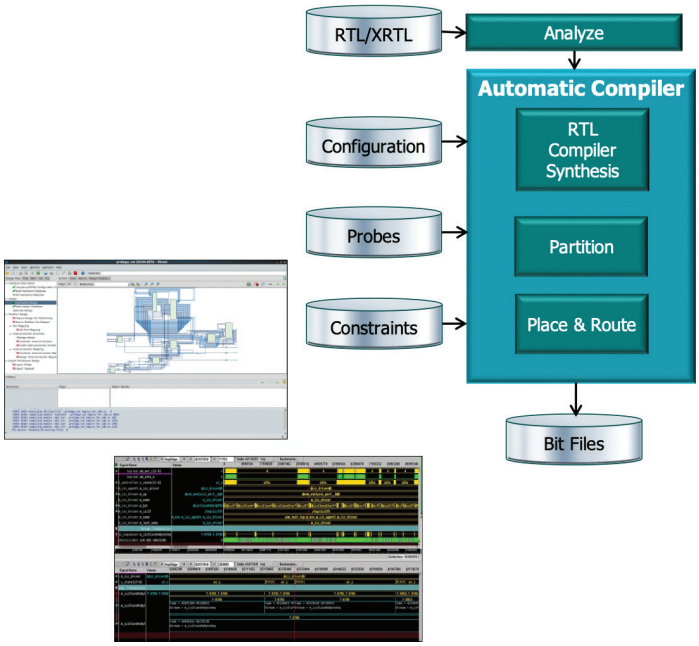


Figure 2. Mapping the ASIC/SoC RTL design into the FPGAs.

Partitioning the design across multiple FPGAs is also addressed. VPS offers automated and guided partitioning of the RTL, freeing users from what can be a daunting task but still giving them the control to apply their unique knowledge of how the SoC is intended to operate to optimize the mapping for maximizing the performance of the prototyping platform.

This combination of mapping capabilities overcomes or reduces the serious challenges design teams face in getting their RTL into the prototype. Together, the capabilities eliminate the need for FPGA experts in the design team, while still achieving prototype operating speeds up to 80 megahertz (MHz) sufficient for booting the OS and running and validating production software prior to any silicon availability.

Visibility and controllability

Building observability and controllability into the design is a perennial problem for teams who work with FPGA prototypes. While both simulation and hardware emulation allow the user to view essentially any signal in the design on any cycle, FPGA prototypes can be far more limiting. In many cases, users must add RTL to the design code to intercept signals of interest and route them to a debug controller.

But this is labor-intensive, so teams tend to only plan for signals they believe are likely to be necessary to the debug process (figure 3). Murphy's law, of course, dictates that these plans will be incorrect. There is always that one more signal that, if you could have seen it, would have solved the puzzle. But intercepting and routing that one more signal could mean a recompile.

Controllability is similarly fraught. To pause and single-step an entire SoC prototype requires a good understanding of the design's clock trees, how they relate to each other and how they have been implemented in the prototype. Even just setting a breakpoint may require understanding what simultaneity means across the various clock regions of the design.

Veloce proFPGA eases these challenges in a number of ways. First, it does automatic insertion of probes–up to 2.5 million of them per FPGA–into prototype signals. So virtually all the signals in most designs will be probed automatically. Second, Veloce proFPGA routes the selected signals through trigger and conditional-capture circuits, through lossless data compression and into a real-time trace buffer.

From there, trace data can be streamed to dynamic random access memory (DRAM) or directly to a host. You can change which signals you want to watch and alter the trigger and trace parameters at run time, eliminating those recompiles. When the prototype involves multiple FPGAs, Veloce proFPGA automatically time-aligns the signals from different chips.

For a more detailed examination, you can stop the prototype clocks with a trigger and then proceed in single-step mode. In this mode, all register and memory contents are accessible. All memory instances are accessible through backdoor ports that allows both reading and writing without disturbing the functional circuitry. The tool will even reconstruct the values of signals within combinatorial circuits between registers if you so require.

This extensive visibility and control make it possible to take on challenging hardware-software interactions. The team can use their favorite software debug environment for code running on the Veloce proFPGA embedded Arm cores or use Veloce proFPGA probes and debug functions to set triggers and trace code executing on processor cores implemented in the FPGA fabric. In this way, they can study both software execution and the operation of the RTL in a synchronized fashion.

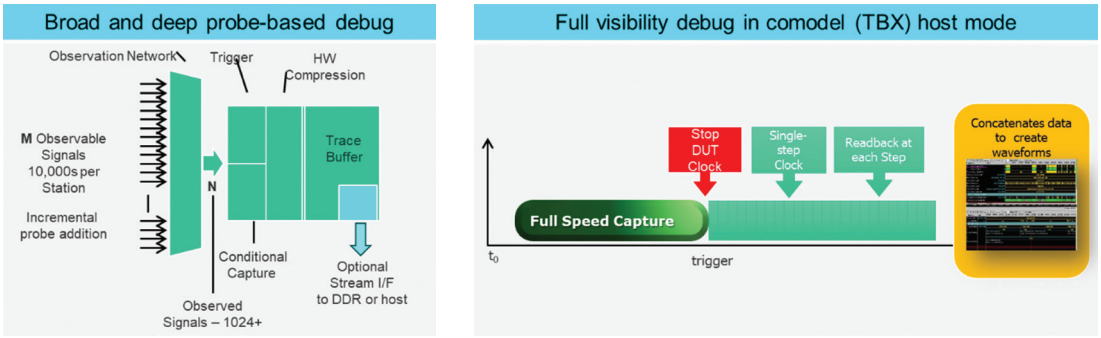
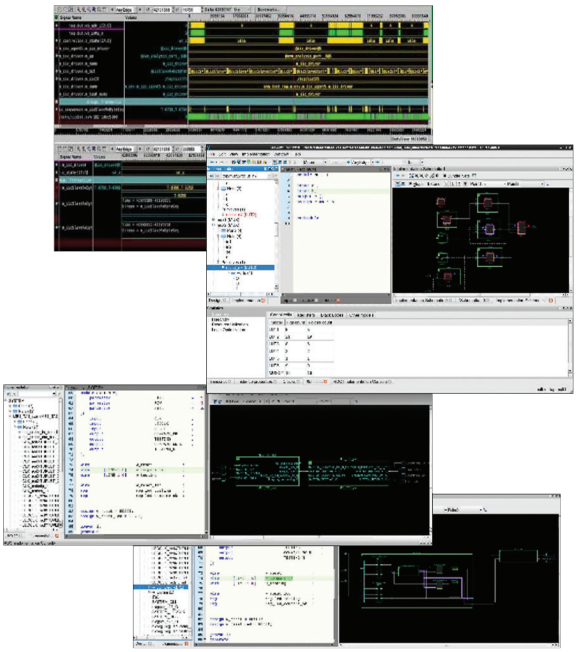


Figure 3. Debug process.

Visualizer

All this observability and control will only be useful if there is a way for humans to make sense of the resulting data. Accordingly, Veloce proFPGA uses the Siemens Visualizer debug environment to manage triggers and traces, present data and explore the RTL design and the FPGA netlists using the original design signal names. The tool gives special attention to making sense of clock nets, combinatorial loops and other inscrutable structures in the FPGAs.

This is the same Visualizer environment used with Questa™ simulation, the Veloce Strato emulation platform and the Veloce Primo enterprise prototyping system, as well as with some formal verification and clock-domain-crossing tools. So team members do not have to change user interfaces as they move from task to task during the design, nor when they share data with other groups.



Practicality

As a complete, expandable system, Veloce proFPGA is designed to be used and managed by verification teams themselves, without recourse to FPGA prototyping experts or tech support groups. Systems can be placed where they are needed. As verification activities expand to include third-party IP or code developers or early customers, Veloce proFPGA systems can be shipped to remote locations with every expectation that they will run correctly out-of-the-box. This can prove a huge advantage over fragmented or one-off prototypes, which can be delicate, complex to bring up or simply not reproduceable outside the lab that built them.

By combining this flexibility with an FPGA prototype's ability to provide near- or at-speed emulation of an SoC design starting early in the design cycle, Veloce proFPGA can be a huge help to IP developers, verification engineers and code developers. By overcoming the challenges that have beset FPGA-based prototypes in the past, this Siemens EDA platform puts remarkable power in the hands of these teams while freeing them from reliance upon FPGA experts or FPGA programming gurus. The Veloce proFPGA platform can be a boon to all manner of SoC development efforts, from early in IP development to the final stages of on-site system integration.

Siemens Digital Industries Software

Americas: 1 800 498 5351

EMEA: 00 800 70002222

Asia-Pacific: 001 800 03061910

For additional numbers, click here.

Siemens Digital Industries Software helps organizations of all sizes digitally transform using software, hardware and services from the Siemens Xcelerator business platform. Siemens' software and the comprehensive digital twin enable companies to optimize their design, engineering and manufacturing processes to turn today's ideas into the sustainable products of the future. From chips to entire systems, from product to process, across all industries, Siemens Digital Industries Software is where today meets tomorrow.

siemens.com/software

85333-D2 5/23 H