FPGA PROTOTYPING SOLUTIONS

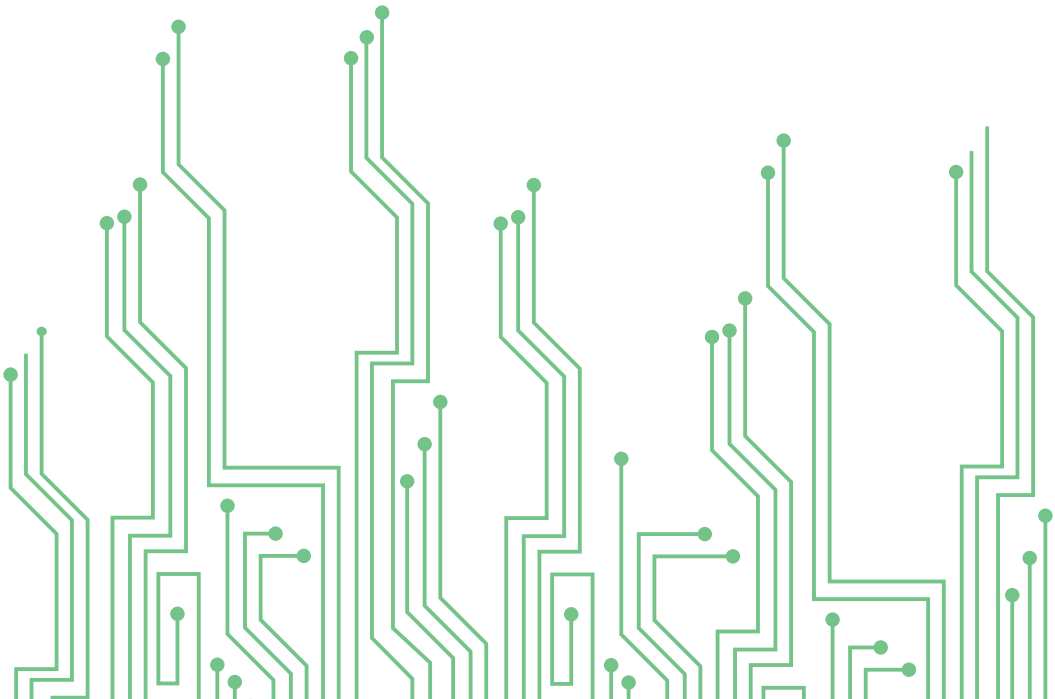
Whitepaper

How do Logic Simulation, Emulation, and FPGA Prototyping work?

Author:
Xusheng Zhang

Release date:
September 2023

Contents



In chip development, design verification holds immense significance. Before the chip enters production, it's imperative to ensure that the design conforms to the specifications and that all potential risks are identified and addressed. These prevent unfixable hardware bugs after tape-out, reducing risks in later stages. As the chip's scale and functional complexity increase, the challenges in verification also intensify. The biggest challenge is ensuring chip design's accuracy and efficiency while reducing verification complexity.

Designs under test (DUT), consisting of the RTL (Register Transfer Level) codes, are the heart of verification. The verification process starts with gathering requirements, then partitioning into subsystem modules and further into functional modules. RTL hardware description language (HDL) files are then generated, serving as the foundation for engineers to construct the Testbench.

Testbench acts as a platform to simulate the verification environment and control the inputs to the DUT, including functional models, input stimuli, and online data interaction, as illustrated in Figure 1. The key to verification is to test DUT and its adherence to the specifications.

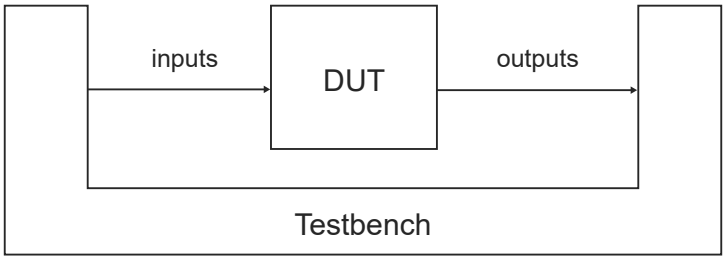


Figure 1. Testbench built by verification engineers

How can we guarantee the DUT's suitable operation with intricate RTL codes? The answer lies in functional verification. The frequently used verification approaches include simulation, emulation, and prototyping. Each of them has its advantages and disadvantages. Therefore, verification methods should be selected according to the requirements of different design stages. The proper adoption helps accelerate verification convergence for efficient production. All these methodologies serve the DUT.

In the following sections, we will delve into the mechanics of logic simulation, emulation, and FPGA prototyping, focusing on their interaction with the DUT.

Logic Simulation

Logic simulation is to mimic and validate the functions and features of digital circuit designs written in hardware description languages (HDL). This process aims to verify whether the circuit design meets its intended purpose by simulating hardware behavior within a computerized environment. Simulation is the key to the accurate execution of the design based on models written in HDL, such as VHDL or Verilog, to test the design's functionalities.

A simplified logic simulation verification system is shown in Figure 2. In this process, the test vector operates within the testbench while DUT and testbench run through the software-based simulation system. The resulting outputs are then compared to the expected outputs.

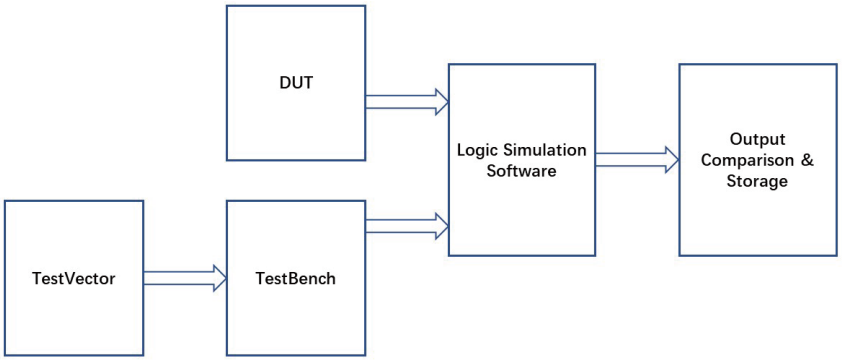


Figure 2. Logic simulation verification system

However, as chip design scales up, test platforms written in Verilog or SystemVerilog could be more efficient for testing. Issues arise concerning communication between fundamental components within the test platform. Other problems may happen during the components' establishment, management, and reusability. Hence, the Universal Verification Methodology (UVM) emerged as a solution.

In the complex system of chip (SoC) design, UVM paves the way to build a robust test platform, capable of processing amounts of design and verification tasks. UVM's significant advantage is its reusability. It allows designers to employ the same verification environment across multiple projects, escalating design efficiency. In addition, as UVM is an industry-standard, its utilization facilitates collaboration with other design teams. Besides, UVM components can also be shared with this common standard. Figure 3 is a typical UVM verification platform diagram.

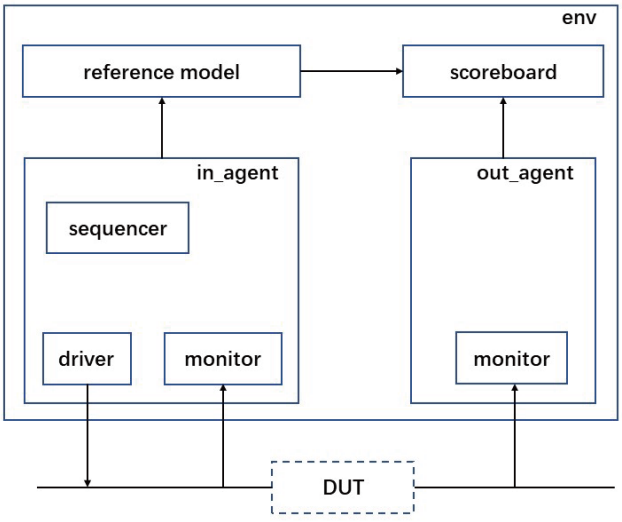


Figure 3. UVM verification platform diagram

Generally, logic simulation is categorized into functional simulation, post-synthesis simulation, and timing simulation. These categories correspond to the stages after the input of RTL design, design synthesis, and place-and-route.

Functional Simulation: Also known as RTL simulation or pre-simulation, it is the first step of simulation verification. The goal is to validate whether the designed functionality conforms to the expectations under ideal conditions. The design's behavior is observed during this phase by simulating its outputs for specific inputs. It's like a 'preview' for the DUT, allowing us to identify the logical errors in the design without physical hardware.

Post-synthesis Simulation: This happens in the simulating process after synthesis. The goal is to confirm that the circuit configuration aligns with the intended purpose. Synthesis tools are employed in this phase to transform HDL codes into a logical netlist. This netlist is later used for simulation, verifying whether the circuit behavior in the post-synthesis corresponds to the design intent.

Timing Simulation: Finally, during the timing simulation or post-simulation, we design the potential timing issues arising from hardware and other processes. These issues include component delays, routing delays, power concerns, and thermal concerns. In this phase, we employ more complex simulation models, including those accounting for delayed information, to accurately simulate hardware behavior.

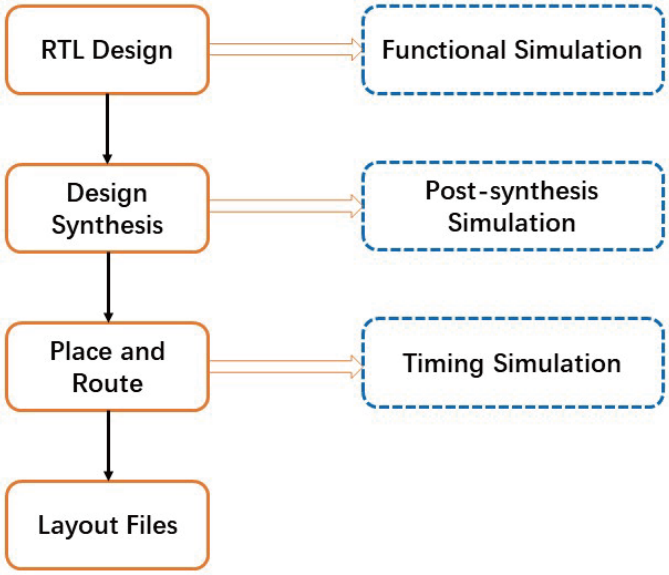


Figure 4. Logic simulation application

In each simulation above, we establish a testbench to manage the inputs and environment of the DUT, and then compare the outputs to our expected results. The shared goal of these three simulations is to guarantee the chip design meets the anticipated functionality and performance throughout the process.

Consider S2C PegaSim, a commercial and mixed-language logic simulator that employs innovative algorithms to achieve a high-performance simulation and constrained random solver. While supporting System Verilog, Verilog, VHDL, and UVM, PegaSim offers back-annotation and gate-level post-simulation, including functional coverage and RTL code coverage. Furthermore, PegaSim's innovative software architecture accommodates diverse processor architectures such as x86-64, RISC-V, and ARM.

While simulation is essential for engineers, current businesses tie the simulating power and computing power to software licenses. Simulation services provided by suppliers are often charged in the form of these licenses. In practical use, engineers often struggle to match the efficient computing power with the calculated demands from the tools. This challenge is illustrated in Figure 5.

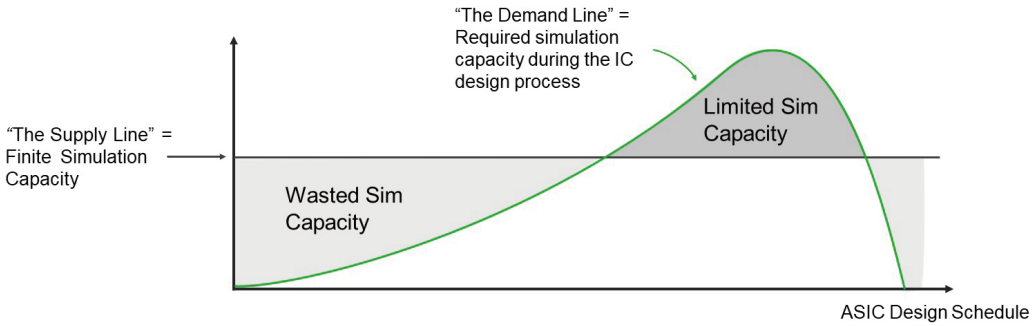


Figure 5. The design and verification resource conundrum

Beyond the conventional licensing model, S2C PegaSim Logic Simulation embraces an innovative business approach by providing a ready-to-use online cloud platform. When the DUT is subjected to regression testing and random coverage, PegaSim will work as shown in Figure 6. This approach caters to diverse corporate requirements, addressing issues including license shortages, insufficient resources, and prolonged license occupation by design engineers. PegaSim enhances verification efficiency by providing on-demand and unlimited simulation capacities.

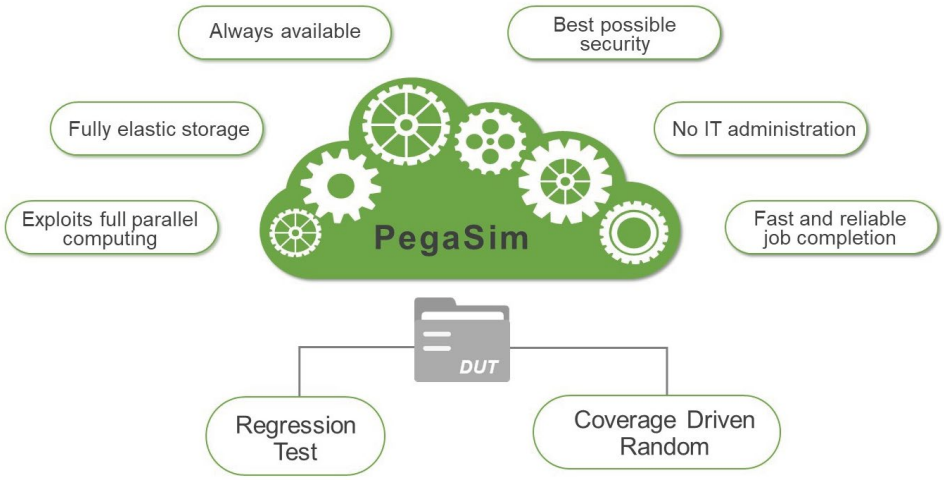


Figure 6. Online simulation cloud platform provided by PegaSim

Emulation

While logic simulation is user-friendly, cost-effective, and offers powerful debugging capabilities, it doesn't perform well in large-scale digital circuit designs. As the design complexity increases, simulation time also grows. Therefore, employing specialized hardware-based methods for chip design debugging, such as emulation and FPGA prototyping, is of great importance.

Emulation outperforms logic simulation in runtime and debugging efficiency. This is attributed to the fact that the emulator can accelerate automatic emulation and enable chip design debugging. Emulation is often employed in the early stages of large-scale SoC designs for RTL functional verification.

During emulation, hardware designs, written in HDL like Verilog or VDL, are initially compiled, and then loaded onto the system. In some cases, these designs are loaded onto specific hardware (e.g., FPGA). Once loaded,

the design starts running in the emulation. Its performance can be observed and debugged with embedded tools. Engineers then will analyze the design based on the results, optimizing it by identifying and resolving issues.

Emulation runs faster than simulation and can mimic real-world hardware behavior. This feature is very helpful in hardware design and verification, especially for complex and large-scale hardware systems. Emulators comprise hardware and software. For instance, consider **S2C OmniArk Emulation**, a hybrid and transactional emulator. OmniArk's hardware consists of multiple FPGAs, extendable to hundreds, while the software contains **Compile, Runtime, and Debug**, as shown in Figure 7.

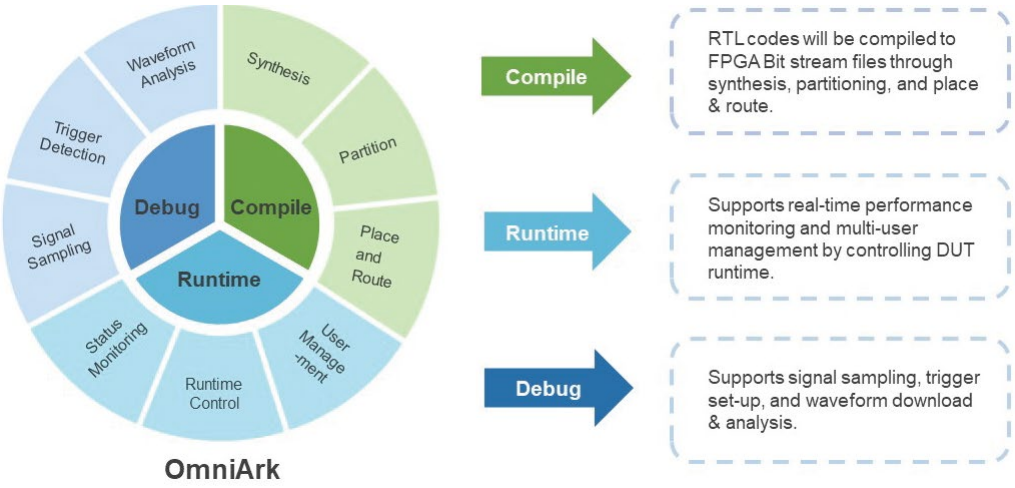


Figure 7. The software of OmniArk Emulation

Compile: In this phase, DUT is mapped onto the emulator via full-automated software for high-speed emulation. The workflow is shown in Figure 8.

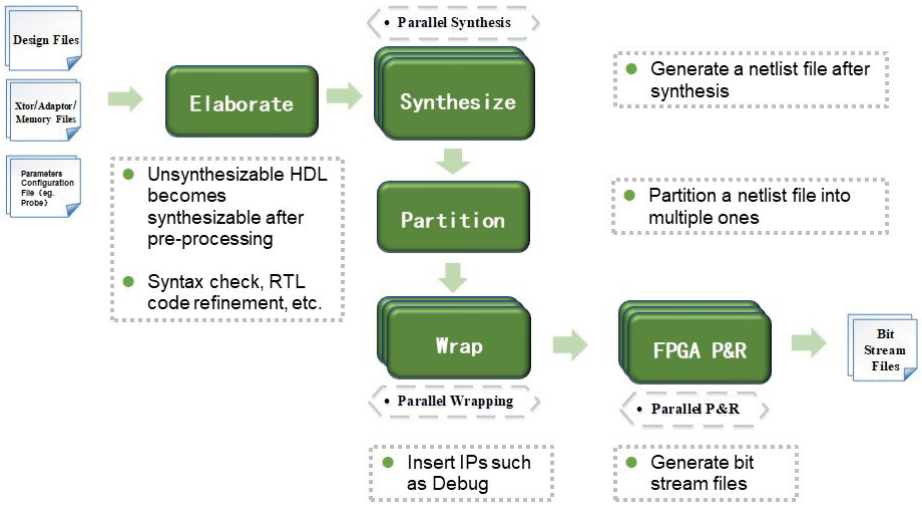


Figure 8. Compile flow in OmniArk Emulation

Runtime: In this phase, the runtime software governs the entire emulation to support various user modes, including ICE (In-circuit Emulation), TBA (Transaction-based Acceleration), and QEMU. Its core components encompass a runtime database, a runtime library, software/hardware interfaces, and user interaction interfaces. The runtime software also enables concurrent use of the devices by multiple users.

Debug: Emulators' debugging capabilities are nearly the same as logic simulators. Emulators also enable full signal visibility through static probes, dynamic probes, and internal logic analyzers (ILA). In addition, signals can be assigned or restored with the ReadBack/WriteBack function.

Furthermore, emulation is equipped with specialized verification intellectual properties (VIP), providing necessary verification interfaces. OmniArk supports APB, AHB, AXI4, AXI4-Stream, AXI4-Lite, UART, SPI, I2C, DDR, Ethernet, USB, PCIe, SPI Flash, NAND Flash, etc. This covers commonly used interface protocols, catering to most verification application requirements. Additionally, S2C can further develop customized solutions according to customer needs.

OmniArk offers a range of innovative software functionalities, including automatic syntactic error correction, Smart P&R, Auto-Block Select (ABS), and diverse information acquisition methods. These features enable MHz-level emulation, full-automatic intelligent compilation, powerful debugging, and other emulation verification methods. OmniArk also boasts an extensive VIP library for hyper-scale system-level verification in high-end general chip design by providing target verification environments.

In conclusion, emulation integrates specialized circuits and logic, with its speed reaching several hundred kilohertz or even megahertz. While functional simulation usually runs at ranging from tens to hundreds of hertz. In comparison, emulation is thousands to hundreds of thousands of times faster than simulation. Therefore, emulation is better in design verification and debugging as it can emulate at higher speeds and provide results rapidly.

FPGA Prototyping

In complex IC design, FPGA prototyping is another crucial verification method. It aims to test and validate the circuit design with prototype hardware resembling the final chip in the early stage. Also, it ensures the correct chip design through a similar real-world runtime. In FPGA prototyping, chip design is first mapped onto FPGAs. Then, the system mimics the chip's functions and application environment to validate the functionality. Through this process, the on-chip software development environment is also provided. As FPGA prototyping runs faster than emulation, underlying software can be developed by software engineers. This hardware-software co-development before tape-out is the highlight of FPGA prototyping.

Below are the key steps of FPGA prototyping, including **design partitioning, post-partition system-level timing analysis, programing and downloading, as well as functional verification and debug.**

Design Partitioning: At the initial stage, the complex design (DUT) needs to be partitioned. As a single FPGA can't accommodate hyper-scale design, the design logic needs to be partitioned into smaller sections with specific tools. Each section is then mapped onto one or more FPGAs. This process requires an improved system performance with minimized cross-FPGA signals for less inter-system path delays. A typical RTL partitioning process is shown in Figure 9.

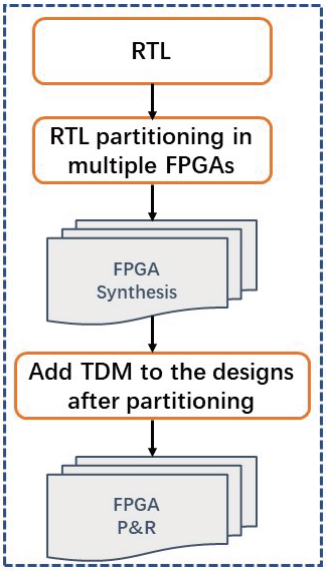


Figure 9. RTL design partitioning flow in Prodigy

FPGA synthesis in the process above refers to transforming DUT into a netlist available in FPGAs. Time-division multiplexing (TDM) in the post-partitioned design is crucial to post-partitioned system performance. Generally, in the post-partitioned design, the interconnect signals between FPGAs far exceed the number of those available among physical connections. Adding TDM means using time-division multiplexing to transmit signals through limited resources. Mapping and P&R refer to mapping the synthesized design onto the specific resources of FPGAs, including look-up tables, flip-flops, and DSP modules, and then placing and routing.

Timing Analysis: Timing analysis ensures the design meets all the timing requirements when running on the FPGAs. When the original user design is partitioned, the partitioned path delays should be considered in timing analysis. These delays, which often reach tens of nanoseconds, primarily come from TDM and inter-FPGA connections. If there are paths do not meet the timing requirements, it may lead to design malfunction. In such cases, methods can be adopted to improve timing performance, such as timing constraint optimization, design optimization, pipeline design, partition boundary adjustment, and place-and-route optimization. These methods help the design up to the expected clock frequency and reduce path delays.

Improving system running frequency is a consideration because it is essential to FPGA prototyping performance. Common approaches are partition boundary adjustment, partitioned results optimization with TDM, place-and-route constraints, and timing-driven partition algorithms. These methods can reduce main path delays and enhance system performance.

Programming & Downloading: After mapping, placing, and routing, the design is compiled into FPGA bitstream files. Then, interconnections are constructed between the FPGAs before the bitstream files downloaded onto the respective FPGA. After download, periphery IPs such as global clock and global reset are configured accordingly. All these ensure that DUT runs correctly on the prototype.

Functional Verification & Debug: This is mainly to test DUT's functionality when running on the FPGAs. We can test the DUT with hardware interfaces or virtual I/O interfaces to validate whether it meets the expected behavior.

Another key consideration in FPGA prototyping is how to debug partitioned designs. In addition to the debugging and monitoring tools embedded in the user design, designers also need to capture the signal waveforms during runtime for analysis. In this scenario, S2C provides MDM Pro, an innovative debug solution, to address the debugging challenges across multiple FPGAs. This solution supports the sampling rate of up to 125 MHz and stores up to 64GB of waveform data.

Taking S2C Prodigy FPGA prototyping as an example, it offers full-automatic partition and compilation with timing-driven RTL partitioning algorithms. Its internal incremental compilation algorithms help achieve rapid iterative version updates, improving the efficiency of development and verification.

As parallel computing is available in FPGA prototyping, more bugs can be discovered by interfacing with daughter cards. Comparatively, the data from stimuli source models in the logic simulation differ from the actual data result, thus failing to cover all corner cases. This is where FPGA prototyping comes in. With FPGA prototyping, we can begin driver development after the approval of SoC's functional validation. We can even provide chip demos to interested customers for preselling before tape-out. FPGA prototyping accelerates product time-to-market by shortening the entire verification time.

Conclusion

By leveraging their advantages and functions, **logic simulation, emulation, and FPGA prototyping** constitute a comprehensive verification solution for chip design. This solution accelerates the chip development cycle while ensuring design accuracy.

Driven by technological development, heterogeneous computing architectures have gradually become the mainstream in chip design. Due to the distinct architectural designs and information processing methods of computing units, verification methods should be adopted accordingly. To expedite the time-to-market, chip design companies have reached a consensus to improve verification efficiency by using different verification tools in different design stages. This strategy has been widely used across various domains in the chip industry.

S2C's heterogeneous verification emerged in this background. It integrates various verification methods, including **logic simulation (PegaSim), emulation (OmniArk), and FPGA prototyping (Prodigy)** to ensure the correct chip design. With S2C's verification solutions, DUT can be fully inspected in terms of function and performance. Its potential problems can also be discovered and fixed to accelerate chip development.