White Paper

Agilex™ FPGAs
Intel® Simics® Simulator

altera™
An Intel Company

Accelerating Embedded Software Development with the Intel® Simics® Simulator for Altera® FPGAs

Creating virtual platforms for Agilex™ 5 SoC FPGA E-Series is now possible using the powerful and proven Intel Simics simulator.

Authors

Kalen Brunham
Principal Engineer Intel

Rolando Santoyo Rincon
Factory Embedded Applications Engineer

Findlay Shearer
Product Marketing Manager

Intel Corp.

Table of Contents

Executive Summary

Computationally intensive workloads – for example, those that employ artificial intelligence (AI) and machine learning (ML) algorithms – are increasingly migrating from data centers to embedded systems at the network's edge. To satisfy the demands of these workloads, the developers of embedded and edge-based systems often employ SoC FPGAs resulting in a sophisticated mix of hardware and software.

An Intel® SoC FPGA includes a hard processor subsystem and hard peripheral functions such as transceivers. Meanwhile, accelerators and other custom functions can be implemented in the device's programmable fabric. The new Intel Agilex® 5 SoC FPGA E-Series deliver the excellent performance per watt characteristics needed to meet the demanding requirements of today's embedded and edge applications.

In a traditional design flow for an embedded system, the main portion of the software development cannot commence until the hardware is available in the form of integrated circuits (ICs), evaluation boards, development boards, and ultimately, the customer's circuit boards. Due to the market pressure to release a final product faster, developers need to start software development as early as possible, even before the hardware in silicon and boards becomes available.

The solution is to create a virtual representation of the hardware called a virtual platform or a digital twin. The virtual platform mirrors its physical counterpart with such fidelity that both can run the same compiled binary software files providing the same results in the software's execution. Through a program called a simulator, the software can be developed, debugged, and verified on the virtual platform up to a year in advance of the physical hardware becoming available.

Intel uses the Intel Simics® simulator as its virtual platform simulator. The Intel Simics simulator has many valuable features that enable the creation of virtual platforms involving multiple processors, network connectivity, serial consoles, and others. The Intel Simics simulator can also be used with virtual platforms featuring models of Intel Agilex 5 SoC FPGAs E-Series to accelerate the embedded software development for systems that use these devices, providing developers with a vehicle to exercise their software similarly to how they would do so with real hardware.

Industry Challenges for Embedded Systems

In the not-so-distant past, one definition of an embedded system was "something you didn't even know existed until it stopped working." Although this may seem a little flippant, it wasn't far from the truth because many embedded systems in the past were implemented as standalone devices with little connectivity to the outside world. These embedded devices were often based on simple 8-bit microcontrollers performing relatively low-level monitoring and control activities on a local device

or system. For example, a small, standalone embedded system might spend its days monitoring the temperature of a machine like a pump, shutting the device down if the temperature exceeded a specified level.

Things have changed dramatically over the years. Today, almost any electronic system that isn't either a desktop computer or a server may be considered an embedded system or include one or more embedded systems. Also, such systems are almost invariably connected – to each other or to a local fog or remote cloud – by wired means, such as Ethernet, or wireless means, such as Bluetooth and Wi-Fi.

Computationally intensive workloads—for example, those that employ AI and ML algorithms – are increasingly migrating from data centers to embedded systems at the network's edge, where localized concurrent processing and analysis are needed to meet demanding system-level latency requirements. At the same time, power is more constrained in many embedded and edge applications, meaning the implementation technology must be power-efficient while providing the required performance attributes.

To satisfy the demands of these computationally intensive workloads, embedded and edge-based systems often employ custom-designed application-specific integrated circuit (ASIC) SoCs or SoC FPGAs. An ASIC SoC includes one or more processors, on-chip memory, hardware accelerators, and various peripheral, communications, and interface functions. An SoC FPGA includes a hard processor subsystem and other hardened functions such as transceivers and external memory interfaces. However, hardware accelerators and other custom functions can be implemented in the device's programmable fabric, providing a more flexible solution than an ASIC.

The software content of embedded systems has increased exponentially over the years. As a result, on average, there are now five software developers for every hardware design engineer on an embedded project.[1] Even when all aspects of hardware development are considered – including architectural exploration, logical design, physical implementation, verification, and more – engineering is currently dedicating more than 50% of the total development effort to software.[1]

A large part of this software development effort is devoted to debugging tasks. On average, embedded software developers spend 20% of their time architecting and designing the code they are going to write, 30% of their time writing the code they just thought about, and the remaining 50% of their time debugging the code they just wrote.[2] What all of this means is that the later embedded software developers manage to start developing, the longer it will take to complete the project.

Common challenges for any company creating embedded systems are that competition is intense, market windows are shrinking, and getting one's product with high-quality software and hardware to the market in a timely fashion is critical to the company's success, or even the company's continued existence in certain cases. Because of these challenges, finding solutions that accelerate the software development process is imperative.

Intel Agilex 5 SoC FPGAs E-Series

ASIC SoCs offer the lowest power and the highest performance. However, they are extremely expensive and time-consuming to develop. Also, any algorithms implemented in the form of hardware accelerators are effectively "frozen in silicon." This is problematic in applications and markets where standards and protocols are constantly evolving, which is the case with most of today's high-end applications and markets.

By comparison, algorithms implemented in the programmable fabric in FPGAs can be quickly and easily modified to address any changes in standards and protocols. Functions implemented in the programmable fabric can execute tasks in a massively parallel fashion, thereby providing high bandwidth while consuming relatively low power. These capabilities are complemented by the hard processor subsystems in SoC FPGAs, which – amongst other tasks – can be used to provide command and control functionality for the hardware accelerators implemented in the programmable fabric.

In 2019, Intel introduced Intel Agilex 7 FPGAs and SoC FPGAs to service the extreme capacity and performance requirements demanded by the network core and data centers. Following the tremendous market acceptance of the Intel Agilex 7 device families, Intel continues to innovate by bringing the high-performance capabilities of Intel Agilex 7 devices into new mid-range Intel Agilex 5 FPGAs and SoC FPGAs. In addition to all the functions provided by Intel Agilex 5 FPGAs, their SoC FPGA counterparts also include a hard processor subsystem (HPS).

All members of the new Intel Agilex 5 device family – including the performance-optimized D-Series and the power-optimized E-Series—are manufactured on Intel 7 technology, have a monolithic construction, and come in smaller sizes, both in terms of numbers of logic elements (LEs) and physical packages, which allows them to deliver the excellent performance per watt characteristics needed to meet embedded and edge application requirements. For these discussions, we will focus on the power-optimized Intel Agilex 5 SoC FPGA E-Series.

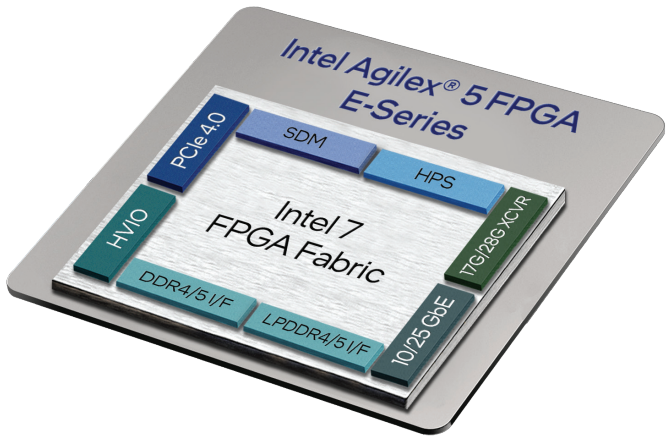

Figure 1. High-level visualization of an Intel Agilex 5 SoC FPGA E-Series.

The programmable logic fabric in Intel Agilex 5 SoC FPGAs E-Series is based on the same architecture used in the high-performance Intel Agilex 7 device family. These devices also feature multiple input/output (I/O) capabilities that can handle nearly any I/O task needed in equipment used in embedded and edge applications, including high-speed SerDes transceivers, high-speed general-purpose I/O (GPIO) banks, high-voltage I/O (HVIO) banks, and hard memory controllers for DDR4, LPDDR4, DDR5, LPDDR5, and QDR-IV SRAM.

The upgraded HPS in Intel Agilex 5 SoC FPGAs E-Series incorporates two 32/64-bit Arm Cortex*-A76 and two 32/64-bit Arm Cortex-A55 processor cores. The Arm Cortex-A76 cores operate at clock speeds as high as 1.8 GHz, while the Arm Cortex-A55 cores operate at clock speeds as fast as 1.5 GHz. In addition to L1, L2, and L3 caches, this upgraded HPS also includes a floating-point unit (FPU) and a system memory management unit (SMMU) that enables system-wide hardware virtualization and supports advanced operating system (OS) implementations such as Linux*.

Intel Agilex 5 SoC FPGAs E-Series also incorporate MIPI* IP to support a wide range of video applications. Furthermore, the hard Ethernet MACs in these devices implement Time Sensitive Networking (TSN) endpoint functionality compliant with the IEEE 802.1AS-2020, Qav, Qbv, Qbu, and IEEE 802.3br standards.

Virtual Platform Value Proposition

The key value proposition associated with virtual platforms is that they allow software to be developed and tested on virtual hardware before physical hardware becomes available.

The virtual platform, along with the Intel Simics simulator, provides access to hardware internals for observation, fault injection, early hardware/software integration, and software debugging, allowing a smooth transition once the physical hardware becomes available. This is made possible because the Intel Simics simulation can run the same software binary as the actual hardware. Using virtual platforms can also bring immediate benefits during post-silicon activities as they can facilitate software maintenance and continuous integration processes without requiring large board farms.

Virtual Platforms, SoCs, and SoCs FPGAs

As previously noted, one aspect of embedded systems based on ASIC SoCs is that these devices take a long time to be created – several years, in some cases. The SoC development process involves the creation of the device itself along with the reference software stack that will be run on this device. The challenge for the developers of these systems is to make the entire development cycle as short as possible. Based on this, the three main facets of an SoC development cycle are hardware, software, and integration and test, where the latter involves ensuring that the hardware and software work together as planned and identifying, debugging, and resolving any problems.

The hardware portion of the SoC involves architecting, designing, and building the silicon chip itself. This is followed by creating evaluation boards and development boards based on this device. The software portion includes porting the bootloader and the OS to the SoC and developing low-level firmware, middleware, libraries, software development kits (SDKs), application programming interfaces (APIs), and application software.

In a traditional development cycle as seen in Figure 2a, software development cannot commence until the first silicon has been returned from the foundry and packaged and verified. Similarly, integration and testing cannot commence until after the software is well-progressed in its development. All of this increases the duration of the development cycle, thereby delaying the product's presentation to the market.[3]

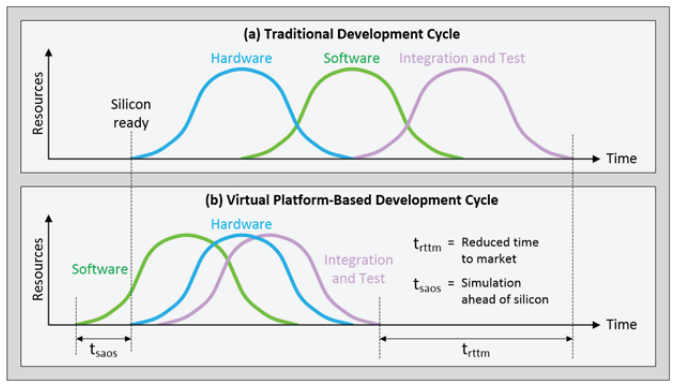

Figure 2. Traditional vs. virtual platform-based development cycles.[3]

In software development, the term "shift-left" is used to describe the concept of moving software development and integration and test so both occur earlier in the development cycle. The way this is achieved is to create a simulation model of the SoC FPGA and to use this model as part of a virtual platform, which allows software development to commence in advance of the silicon becoming available. This is seen in Figure 2b.[3]

Many people think of SoC FPGAs as being readily available "off-the-shelf" parts. On this basis, they may question the need for the use of virtual platforms with these components. However, ease of access to these devices is true only after they are in full production. Consider a new component, such as the Intel Agilex 5 SoC FPGA E-Series, for example. These devices won't move into full production until 2024. Early access to a simulation model and corresponding virtual platform means that software developers can commence work anywhere from nine to twelve months in advance of a new FPGA becoming physically available. Even when a new family of FPGAs becomes available, their corresponding evaluation and development boards usually lag by one or two more quarters. This reinforces the need to employ virtual platforms with SoC FPGAs.

Virtual platforms can also be used in post-silicon activities as part of the software continuous development and integration process, supported by the automation capabilities of the simulators. During this stage in the development cycle, even after the hardware is available, the physical boards are typically limited and must be shared among all members of the development and validation teams, creating a bottleneck in the process. This constraint doesn't apply to the use of a virtual platform since each team member can have their own simulator installation on their own computer, thereby providing an additional advantage for using the virtual platform.

Virtual platforms are not expected to fully replace the need for testing on physical hardware. Instead, virtual platforms are used for the first level of continuous integration loops to provide fast feedback to developers. Once testing proves successful on the virtual platforms, it migrates to real hardware. As illustrated in Figure 3, this strategy implies that the virtual platform modeling needs to be precise enough to support positive and negative testing. Otherwise, errors may remain hidden.[4] This strategy reduces the use of physical boards, thereby diminishing the bottleneck caused when this resource is shared.

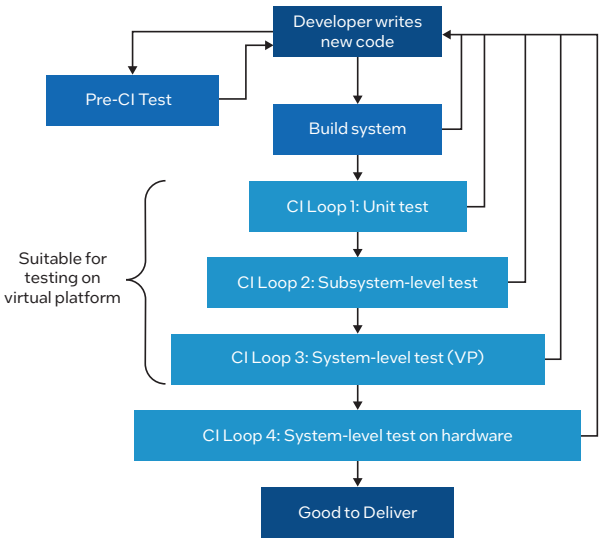

Figure 3. Continuous integration flow using virtual platforms.[4]

A key aspect of simulation is that it often provides better "visibility", meaning the ability to observe the contents of registers and other signals, coupled with better "controllability", meaning the ability to inject faults, into the virtual platform than is available in the real world with a physical system.

Virtual platforms of this type can be used by software developers up to a year ahead of the silicon becoming available. In turn, this results in a significantly reduced time to market since the software development can be started earlier. According to a study by the IBM Systems Science Institute,[5] the cost of fixing defects increases dramatically as they move further along in the development lifecycle. As well as speeding the availability of the software, one of the main advantages of shift-left integration and testing is that it allows bugs to be detected and addressed as early as possible in the development process. Furthermore, testing and debugging the software on the virtual platform greatly eases the act of transitioning to the real system when it eventually becomes available.

The Intel Simics Simulator

The Intel Simics simulator is a full-system, cycle-accurate simulator that provides the technology to build fast, virtual platforms that can run the same production binaries as real-world physical hardware. The Intel Simics simulator is used extensively for pre-silicon and post-silicon software development, testing, and system integration at Intel and by Intel's customers and partners.

As illustrated in Figure 4a, a virtual platform may involve only a single target machine. This simulated hardware could be a standalone processing device such as a microprocessor, application processor, microcontroller, ASIC SoC, or SoC FPGA. Alternatively, the virtual platform may embrace a circuit board carrying the processing device along with other components such as memory devices (e.g., SRAM and DDR), storage devices (e.g., SD card, NAND, and QSPI), Ethernet ports, and communications devices (e.g., I2C, I3C, and USB), and others.

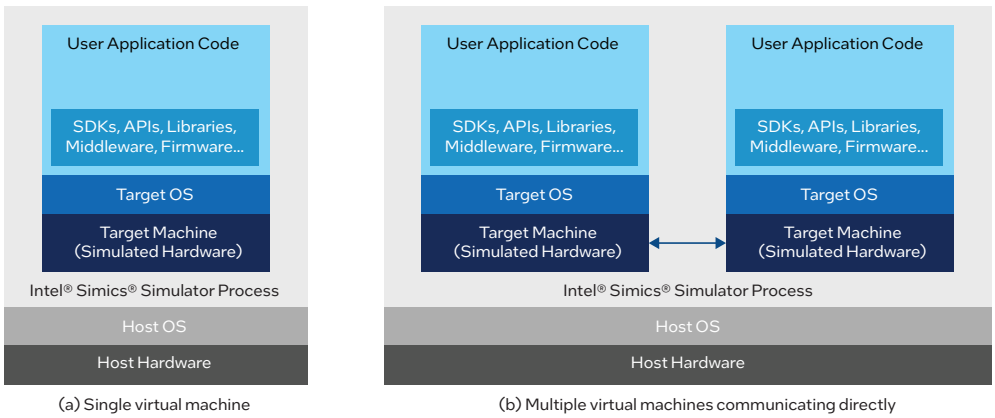

Figure 4. Using the Intel Simics simulator to create virtual platforms representing individual or multiple machines.

In fact, a virtual platform may involve multiple target machines that communicate with each other via GPIOs and communications interfaces such as I2C, I3C, and USB as illustrated in Figure 4b. Yet another possibility is for multiple target machines to communicate with each other and with the host operating system by means of a simulated network as illustrated in Figure 5. The Intel Simics simulator supports the DHCP, DNS, FTP, and TFTP network protocols, and its service node allows it to connect to real networks using forwarding ports.

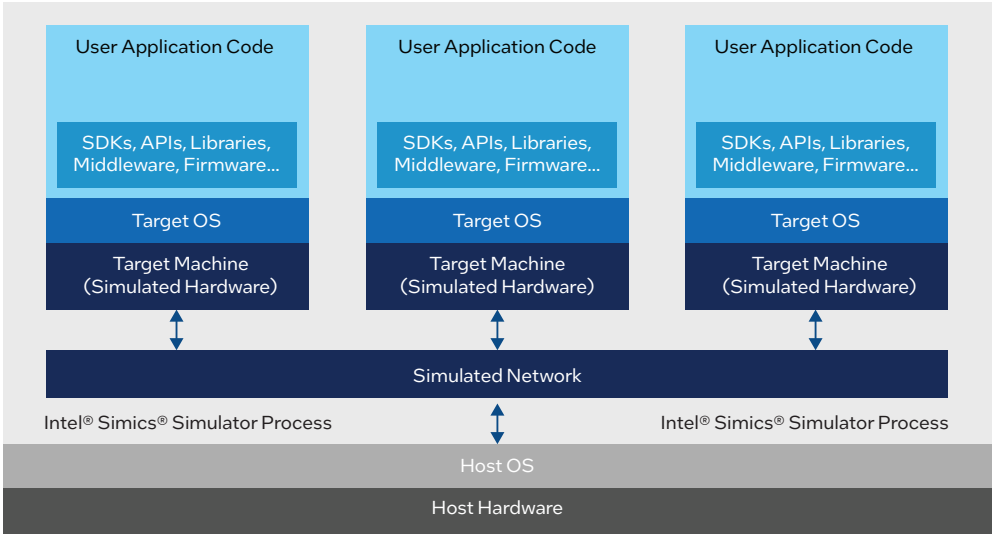

Figure 5. Using the Intel Simics simulator to create a virtual platform representing multiple machines connected via a simulated network.

The Intel Simics simulator is provided with the Simics command line interface (CLI) which is a text console interface that allows the user to control the simulation flow and interact with the simulator through supported commands. Alternatively, the Intel Simics simulator also supports the Ashling RiscFree* Integrated Development Environment (IDE) that facilitates the debug task on the target software from a graphical user interface (GUI).

Once the integration of the software and hardware has been verified on the virtual platform, the same embedded software binaries – including bootloaders, the OS, and user applications – can be loaded onto the physical platform when available, as seen in Figure 6.

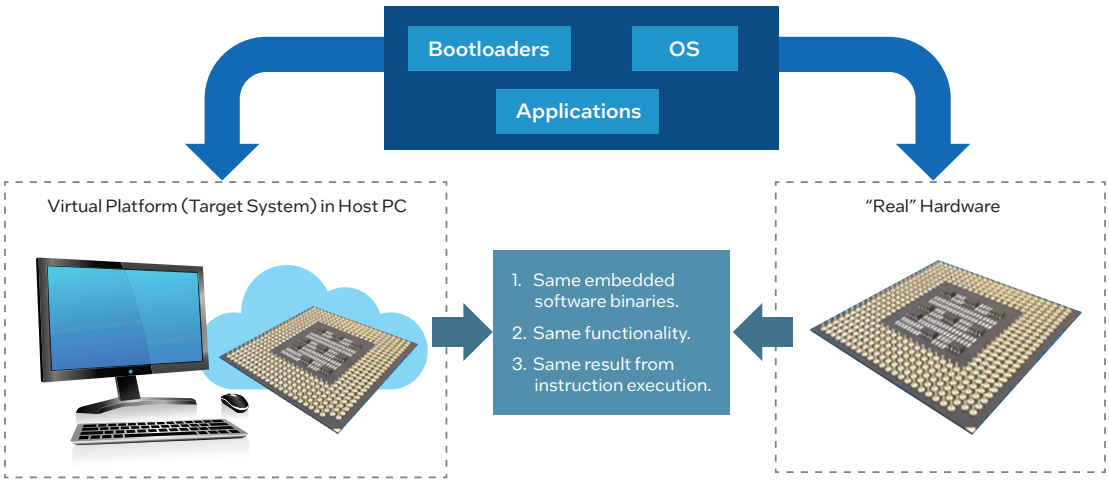

Figure 6. The same software binary files can be run on both virtual and physical platforms.

The Intel Simics simulator supports automation tasks through scripting. This also provides some powerful debug capabilities that facilitate the debug of software problems since this allows users to see the system as an open box.

The Intel Simics Simulator for Intel FPGAs

The Intel Simics simulator for Intel FPGAs supports running on a Linux system host PC along with the Ashling RiscFree IDE. It is shipped today with the Intel Agilex 5 SoC FPGA E-Series Universal Virtual Platform, which instantiates the Intel Agilex 5 SoC FPGA E-Series device Simics model that can run the Linux OS and the Zephyr* RTOS, along with the Arm Trusted Firmware bootloader. Intel plans on working with its ecosystem operating system vendor partners to provide additional OS offerings that can be simulated using the Intel Simics simulator in the future.

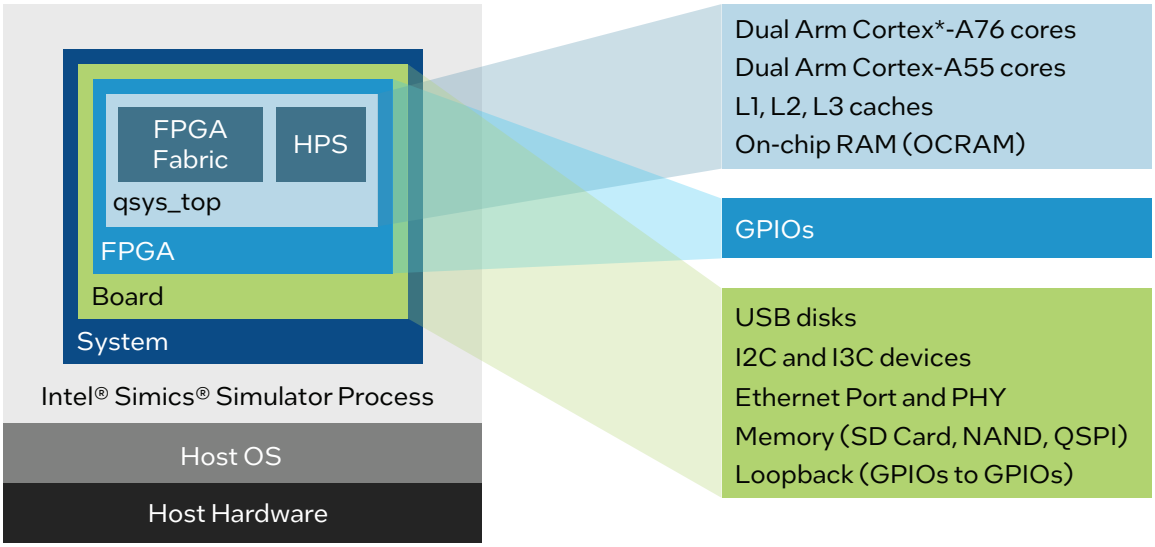

Figure 7. Simics virtual platform for Intel Agilex 5 SoC FPGAs E-Series.

At the time of this writing, the Intel Agilex 5 SoC FPGA E-Series Universal Virtual Platform includes the following components as illustrated in Figure 7:

- **HPS**: A model of the Intel Agilex 5 SoC FPGA E-Series HPS. This component also models all the components in subsystems that integrate the HPS.
- **qsys_top**: This is related to the view of the design that is being modeled and corresponds to the view seen from the Intel Platform Designer under the GHRD. Under this component are instantiated other components such as the HPS (if available), FPGA fabric design, other individual IPs, etc.
- **FPGA**: Top-level view of the hardware design (GHRD) that is being modled from the FPGA device perspective corresponding to the Intel Quartus Prime project. This instantiates the qsys_top component.
- **Board**: A model of a board that contains an Intel Agilex 5 SoC FPGA E-Series. This model integrates the FPGA model with board components such as flash devices, Ethernet PHY, and connectors.
- **System**: A model that represents the complete system. A system model can contain one or more board models.

The virtual platform includes a wrapper called target script, in which the system component is instantiated. In this script, the values of user-configurable parameters are defined. Also, any initial simulation setup and any automation task is performed here.

A more detailed block diagram depicting the architecture of the Intel Agilex 5 SoC FPGA E-Series Universal Virtual Platform is shown in figure 8.

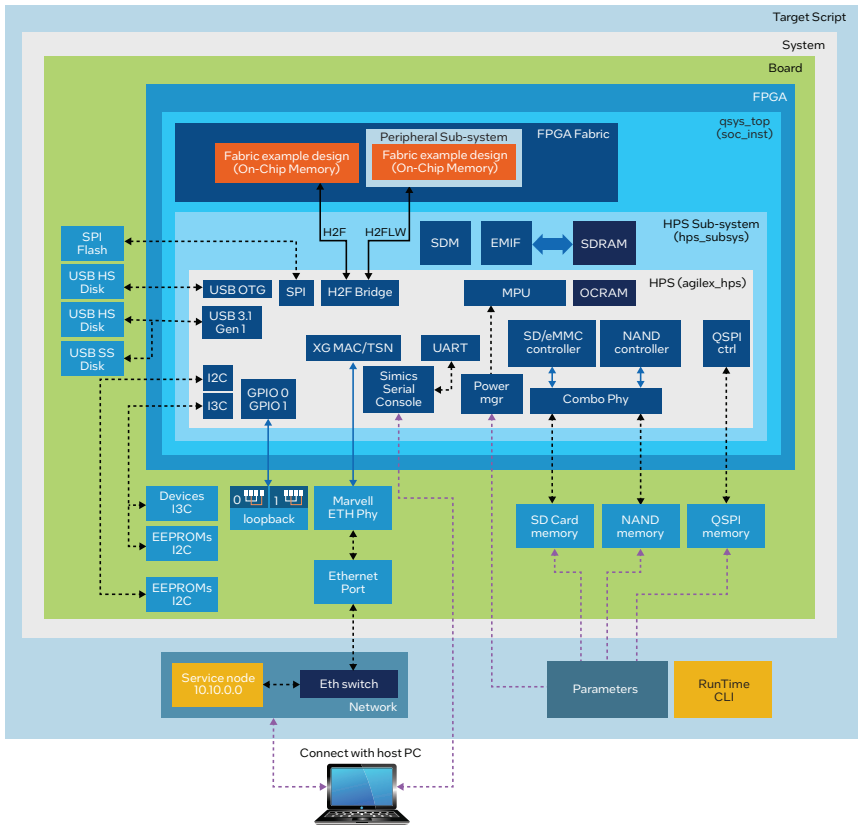

Figure 8. Intel Simics simulator virtual platform for Intel Agilex 5 SoC FPGAs E-Series.

A Simulation Example

Figure 9 shows an example of an Intel Simics simulation with the Intel Agilex 5 SoC FPGA E-Series Universal Virtual Platform using the Ashling RiscFree IDE. In this simulation, the virtual platform is being configured to boot from an SD card using binaries that exercise the U-Boot SPL -> Arm Trusted Firmware -> U-Boot -> Linux flow. When the simulation reaches the Linux prompt, a user mode application written in the C programming language is executed.

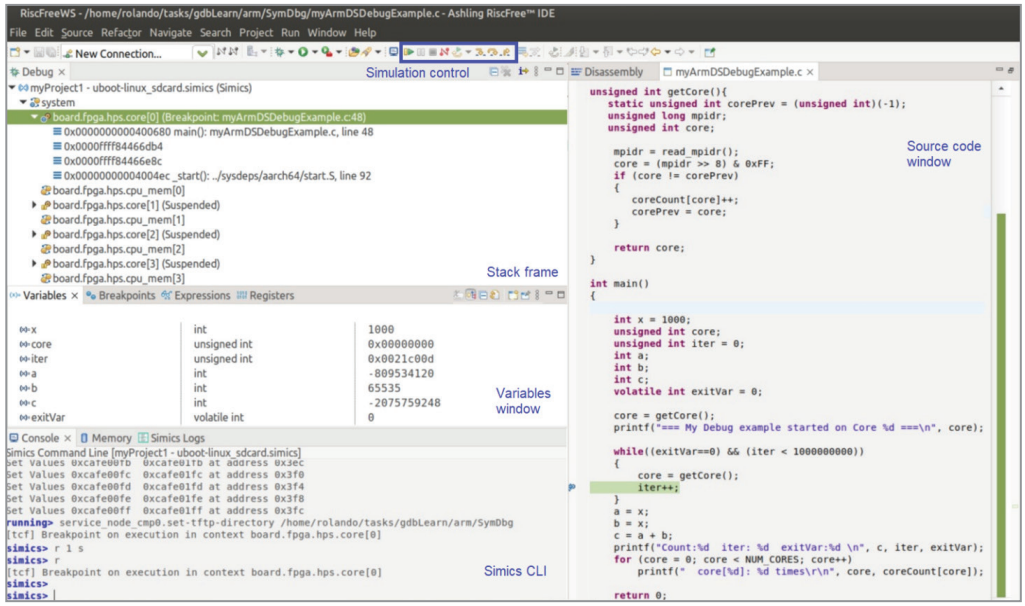

Figure 9. Intel Simics simulation using RiscFree IDE

The RiscFree IDE can be used to debug the application by loading the corresponding symbols file. Figure 9 shows some of the windows in the debug view of the RiscFree IDE. Here, it can be observed that the simulation has hit a breakpoint showing the line in the source code at which it stopped. The IDE also shows the stack frame along with a variable window that allows the user to observe the current values of the variables used in the application. The RiscFree IDE allows the simulation flow to be controlled through standard step-in/step-out/step-over/run buttons along with access to the Simics CLI to enter commands to control the simulation and support the debug process.

Figure 10 shows the output of the serial console associated with the same simulation. This reflects the final portion of the Linux boot process in which we are logging in as **root**, after which we bring the application to the file system using the **tftp** command and then execute the application.

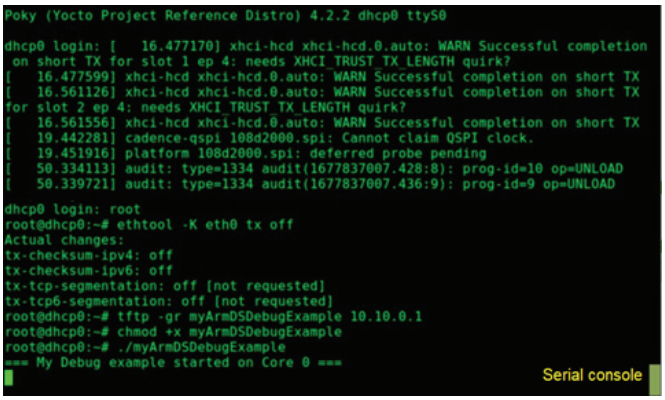

Figure 10. Serial console from the Intel Simics simulation

At the time of this screen capture, the application has commenced its execution and displayed an initial message. This illustration demonstrates that the output observed in the serial console is the same as the one we would see with real hardware if this were already available.

Conclusion

Virtual platforms enable the development and testing of ASIC SoC and SoC FPGA software before silicon exists and before testing on physical hardware.

The use of virtual platforms for ASIC SoC designs is common practice at large engineering companies such as Intel. With the release of the Intel Simics simulator for Intel FPGAs, developers targeting Intel Agilex 5 SoC FPGAs E-series can leverage this industrial strength technology to begin developing and testing their software up to a year before hardware is available, thereby enabling a "shift left" in the development process as compared to a strict waterfall development model.

Learn More

- Simics Simulator for Intel FPGA User Guide
- Simics Simulator for Intel FPGA Intel Agilex 5 E-Series Virtual Platform User Guide
- Linux GSRD Simics Virtual Platform for Intel Agilex 5 SoC FPGAs E-Series Rocketboards page
- Zephyr GSRD for Intel Agilex 5 SoC FPGAs E-Series Simics Virtual Platform Rocketboards page

References

1 Software and System Development using Virtual Platforms. Daniel Aarno, Jocob Engblom

2 https://bit.ly/3JXNX3a

3 D. Aarno and J. Engblom, Software and system development using virtual platforms: Full-system simulation with Wind River Simics, Waltham, MA: Elsevier, Morgan-Kaufmann, 2015

4 Kalen Brunham and J. Engblom, Challenges and Solutions for Creating Virtual Platforms of FPGA and SASIC Designs, Design and Verification Conference and Exhibition Europe, Munich, Germany 2022

5 IBM Systems Science Institute (2004). The impact of defects on productivity. IBM.

altera™
An Intel Company

WP-01329-1.0