WHITE PAPER

Introduction

Most of today's largest semiconductor devices are highly complex system on chip (SoC) designs, which means that they include one or more embedded processors. This indicates that software provides some of the key functionality of the chip. The system cannot be fully verified or validated without both hardware and software. However, software development generally

synopsys.com

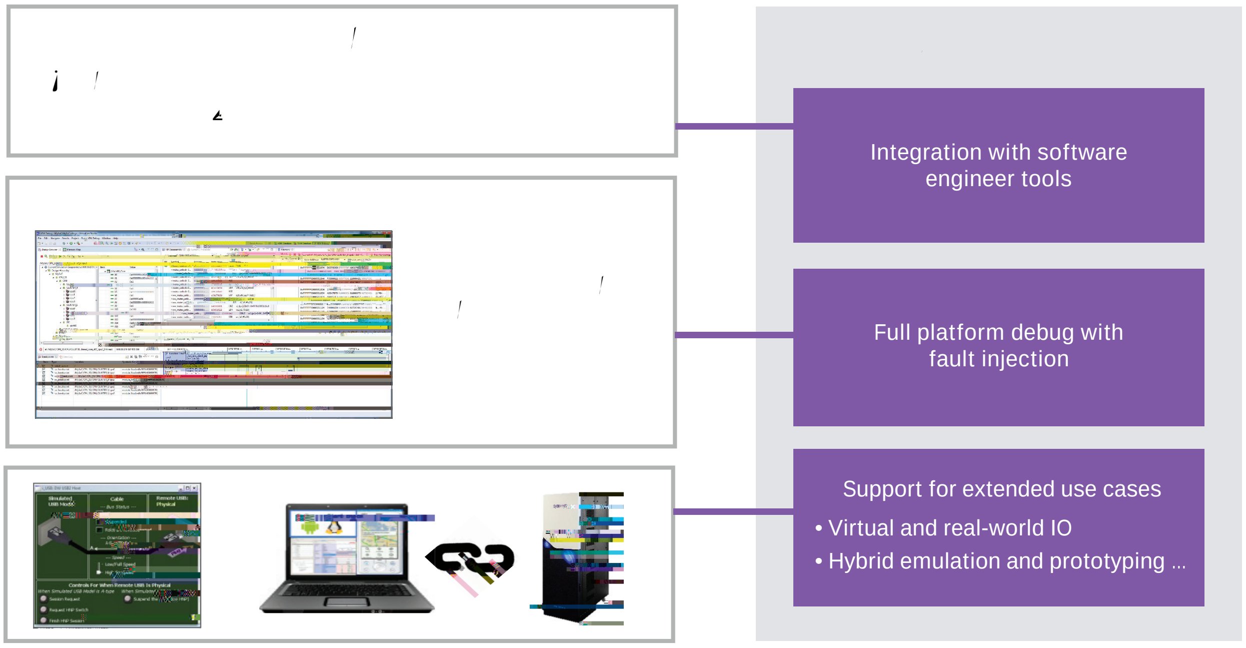

Integration with software engineer tools
Full platform debug with fault injection
Support for extended use cases
• Virtual and real-world IO
• Hybrid emulation and prototyping ...

Figure 2: Key VDK features and links to other tools

Virtualizer Studio supports device virtualization using VirtIO, the standardized open interface for simplified devices such as disks, networks and graphics processing units (GPUs). By replacing the detailed device drivers with the VirtIO models, higher level software such as apps runs faster in the virtual prototype. The industry provides standard models for several types of devices, and these are provided as part of Virtualizer Studio. Example models include VIRTIO_BLK, which implements a block device such as a disk drive and uses a file on the host system for the data, and VIRTIO_INPUT, which implements human input devices such as keyboards and mice. It is possible to implement a virtual LCD touchscreen using the keyboard and mouse of the host system.

Since graphics processing consumes many processor cycles, perhaps the most interesting virtual model is VIRTIO_GPU. It offloads OpenGL commands to the host machine's GPU providing a significant performance boost when running graphics intensive operating systems such as Android and applications with high GPU requirements, for example, by running 2D and 3D graphics applications like Angry Birds on a virtual prototype in near real time. Figure 3 shows an example of how this works. The OpenGL calls from the app are handled by the Direct Rendering Manager (DRM), the part of the Linux kernel responsible for interfacing with GPUs. The Linux virglrenderer then interacts with the host GPU to provide hardware-accelerated OpenGL to the virtual prototype.

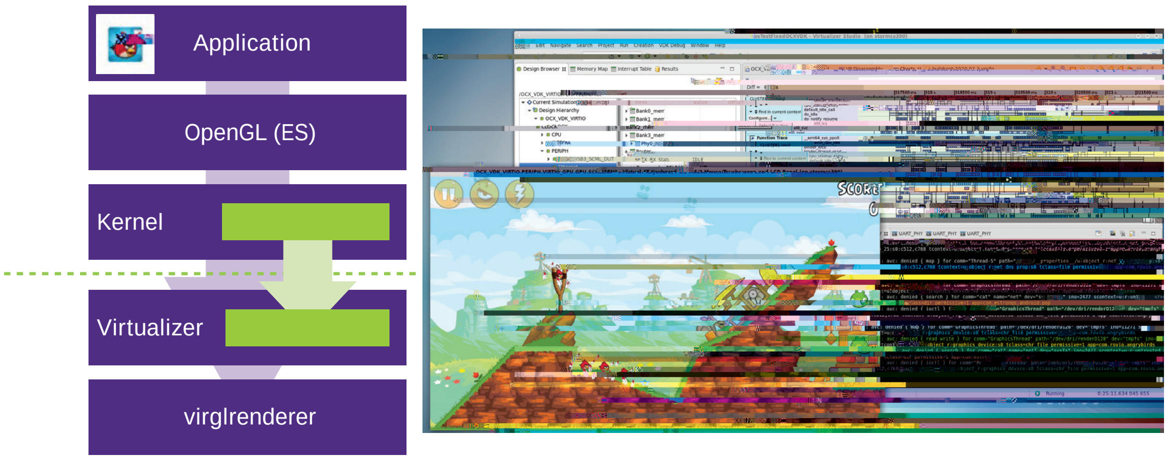


Figure 3: Example of VIRTIO_GPU usage

Industry-Leading Features

Checkpoint Restore

Synopsys Virtualizer Studio provides several unique features that make development and use of virtual prototypes both easier and more flexible. One example is the ability to checkpoint and restore the state of a platform test run. Users can select interesting points to save the current state and then later recover from that same state and continue the run. This supports several valuable use cases:

- Skipping past long initialization phases
- Enabling more tests to be run in a test cycle
- Jumping ahead to points of interest
- Saving error conditions and quickly reproducing them
- Tracking down failures in long runs

Users may set up their runs to take regular snapshots of the system state during long tests. If a failure occurs, the saved states can be used during debug to accurately locate the point of the failure and to quickly get back to the state of the system just before the failure occurred. Figure 4 shows the setup to take advantage of the checkpoint/restore capability.

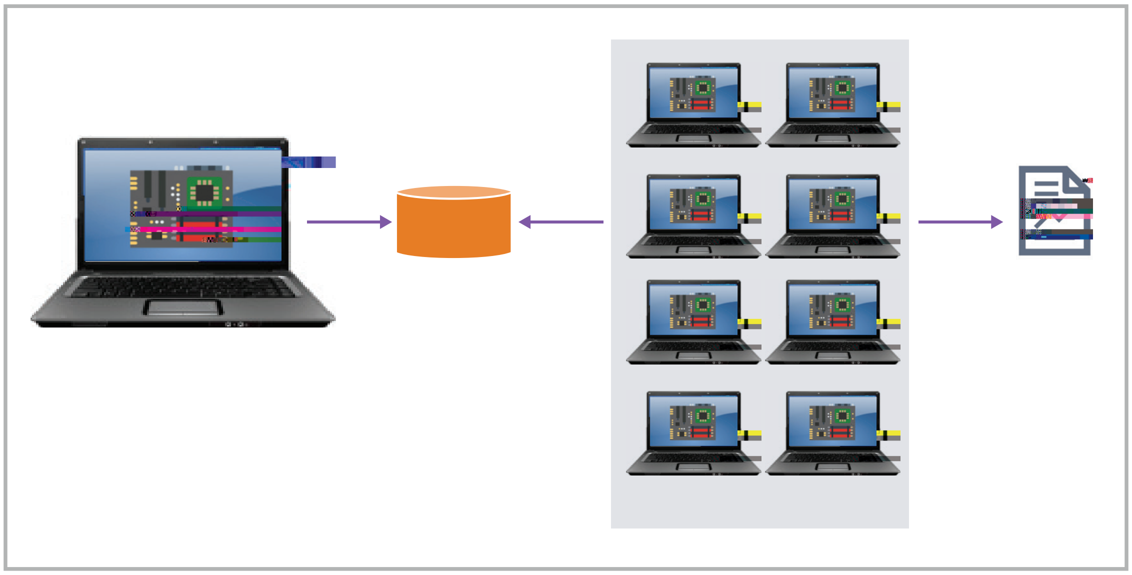
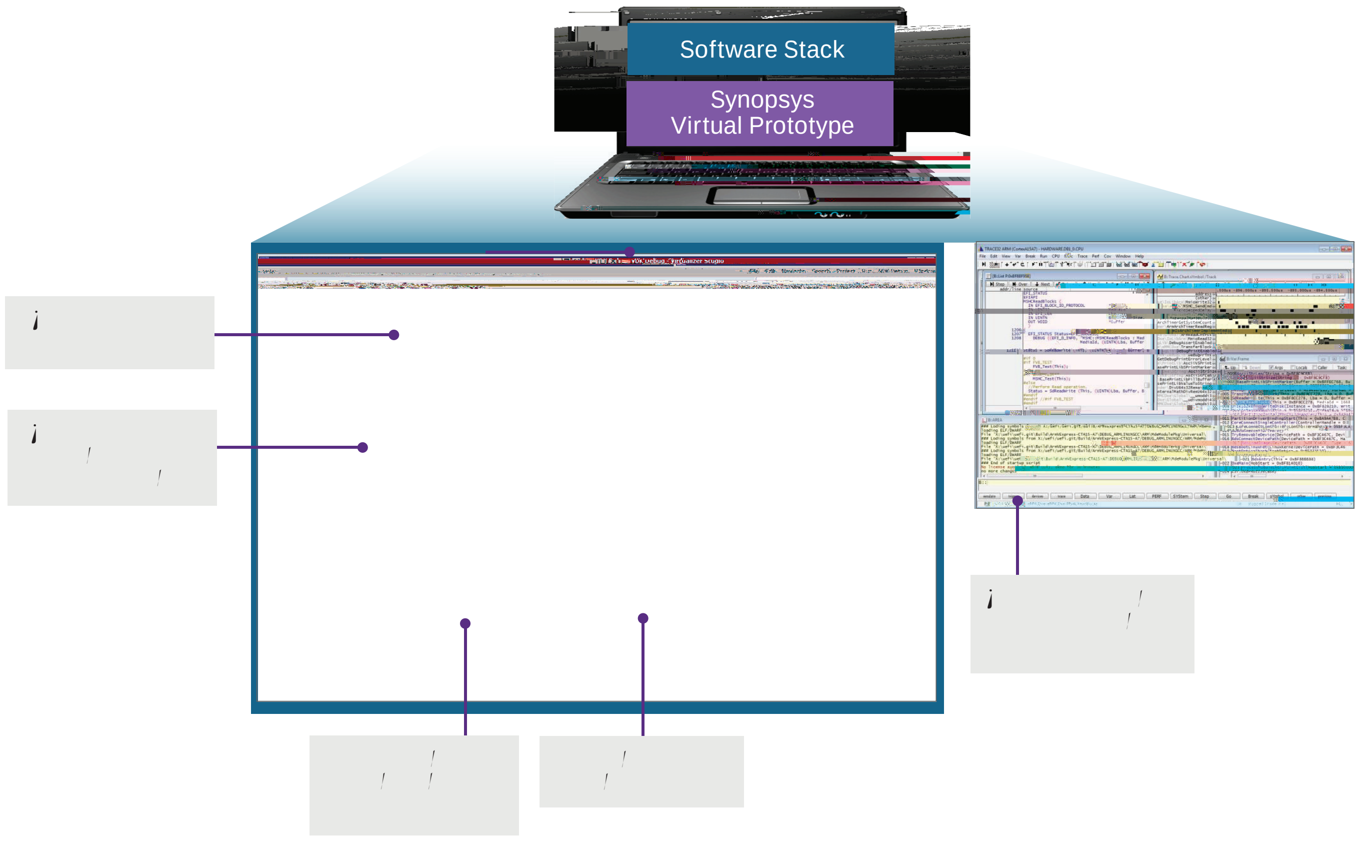


Figure 4: Checkpoint and restore in Virtualizer Studio

Efficient Software Debugging

As mentioned earlier, Virtualizer Studio includes a specialized built-in debugger that is extremely helpful when creating or using VDKs. VDK Debugger has unique features targeted at efficient co-debug of software and VDKs. It monitors activity within both the hardware models and the software, providing the ability to view and analyze both together, correlated in time. It handles both virtual and real-world I/O devices, offers integration with the popular GNU Debugger (GDB) and integrates with other commercial debugging tools. Figure 5 shows just a few of the features that enhance software development productivity on virtual prototypes.


Software Stack
Synopsys
Virtual Prototype

Fast Track Bug Detection

A final unique feature of Virtualizer Studio is Fast Track, which is built into VDKs to quickly detect bugs in configuration or usage. This is invaluable in detecting and diagnosing situations when the host software is not following the correct procedure for model operation. For example, if registers are programmed in an order that violates the specification or if registers are programmed at an inappropriate time, Fast Track traces point this out. Fast Track reports configuration violations, warns of potential fault conditions and indicates the software routine that is behaving incorrectly. This allows the software team to debug the problem quickly, pinpoint the issue and take steps to fix it. Fast Track also provides unique visibility into VDKs by tracing the details of operation. This supports efficient debug of issues found during end-to-end software testing, from apps down to hardware dependent software.

Support for CI and CD

Because continuous integration and continuous deployment are becoming widely adopted by programmers, it is important that any virtual prototype used for software development supports these techniques. CI is the practice of merging all working copies of the software into a shared mainline codebase several times a day. This accelerates development by reducing the effort needed to merge and synchronize the updates from multiple programmers. The integration process includes quality metrics such as code analysis, coverage and performance. A tool such as Jenkins automates the process of building and testing the code while providing a path to CD. If the software is in a constant state of being available for release to customers, then deployment can be automated as well, as shown in Figure 7.

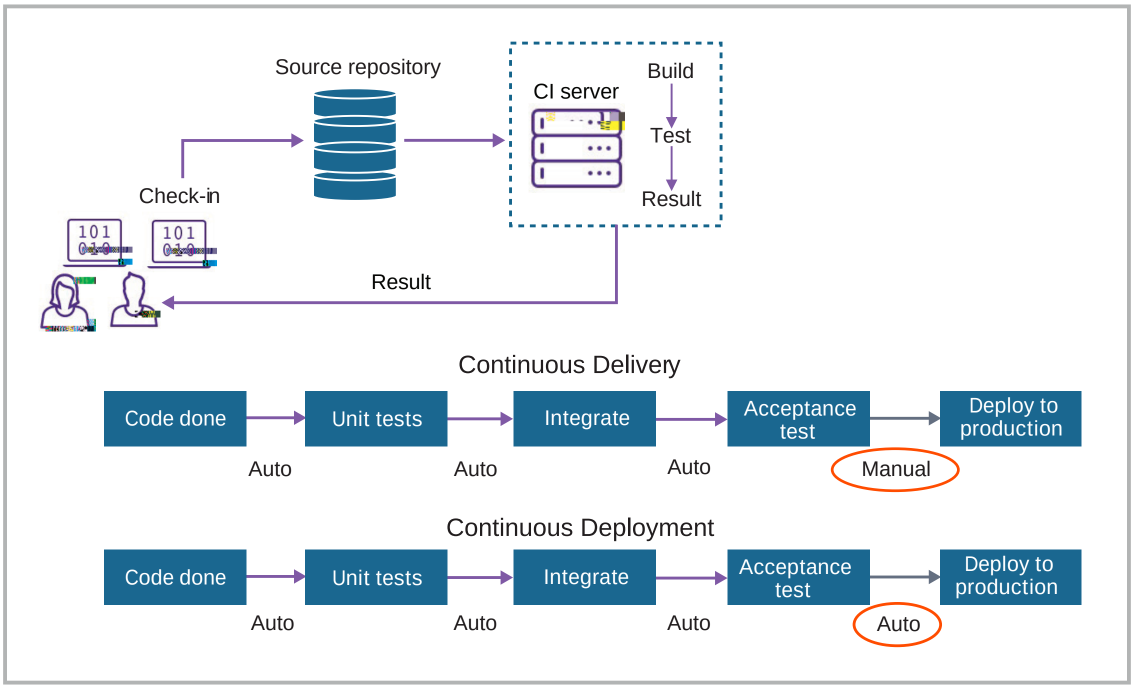


Figure 7: Continuous integration and continuous deployment

Virtualizer Studio supports CI and provides much of the automation needed. Python application programming interfaces (APIs) are available for all features, so tool operations can be scripted. Software build and test systems can use Python scripts to integrate with Virtualizer Studio, for example to configure, start and cleanly shut down a test run. This makes it possible to set up regression runs using virtual prototypes, including cloud deployment. Synopsys provides specific APIs to support regression runs and CI flows. Since Virtualizer Studio is based on Eclipse, it is easy to install plug-ins and to integrate with build and test tools, including Jenkins, and with