ABSTRACT

The ability to restore a virtual platform from a previously saved simulation state can considerably shorten the typical edit-compile-debug cycle for software developers and therefore enhance productivity. For SystemC based virtual platforms (VP), dedicated checkpoint/restore (C/R) solutions are required, taking into account the specific characteristics of such platforms. Apart from restoring the simulation process from a checkpoint image, the proposed checkpoint solution also takes care of re-attaching debuggers and interactive GUIs to the restored virtual platform. The checkpointing is handled automatically for most of the SystemC modules, only the usage of host OS resources requires user provision. A process checkpointing based C/R has been selected in order to minimize the adaption required for existing VPs at the expense of large checkpoint sizes. This drawback is overcome by introducing an online compression to the checkpoint process. A case study based on the SHAPES Virtual Platform is conducted to investigate the applicability of the proposed framework as well as the impact of checkpoint compression in a realistic system environment.

DOI: 10.4018/978-1-4666-2776-5.ch012
INTRODUCTION

Virtual platforms (VPs) are executable software models of hardware systems. The software running on top of the VP cannot tell the difference between the VP and the real hardware. Hence, VPs enable to simulate the complete system long before a first hardware prototype is ready. Compared to a hardware prototype, VPs exhibit a superior observability with deterministic execution and non-intrusive debugging capabilities, VPs form an ideal tool for fast and efficient software development.

Due to the growing complexity of the software being executed on modern systems-on-chip (SoCs) VPs have gained a lot of interest in industry and academia over the last years. Of course, the usability of a VP heavily depends on the achievable simulation speed. Only simulation environments that are sufficiently fast to simulate large workloads can be employed to efficiently develop software. It can be foreseen, that in future more complex multi-processor systems-on-chip (MPSoCs) will be deployed and, hence, there is a strong need to increase the simulation performance of VPs.

The checkpoint/restore (C/R) technique offers the possibility to reduce the number of simulation runs by storing the state of the simulation to a file, and to use this checkpoint image to restart the simulation in exactly the same state at a later point in time. For example, instead of booting the operating system (OS) each time an application is executed, a checkpoint permits to omit the time consuming boot process and to concentrate on the application execution itself. Although the C/R technique does not speed up the VP simulation itself, this technique can be used to reduce the time spent simulating.

For modeling VPs, SystemC (SystemC, 2005) has been established as the de-facto standard and it is supported by all major vendors of VP modeling environments. It allows to efficiently describe software as well as hardware aspects of VPs. The most important use cases for the C/R feature for VPs are listed:

- **Time Saving**: By restoring a checkpoint a certain simulation state can be reached without the need for lengthy simulation runs.
- **Move Around in Time**: The facility of creating multiple checkpoints at different points in time enables the user to move around in time.
- **Periodic Checkpointing**: Periodic checkpointing allows the user to quickly recover the state shortly before an error has occurred.
- **Simulation Transfer**: A developer of a VP can transfer a checkpoint image to another developer to request help for a specific problem or to demonstrate a certain behavior of the simulation.

How the C/R mechanism can be used to accelerate the edit-compile-debug cycle is clarified in the following example of a development task typically occurring when porting an OS to a new HW platform: Debug and analyze how an application running on an Android based smartphone interacts with the hardware after a touchscreen event occurred. This task requires a complete boot of the guest OS on top of the VP. Furthermore, the application has to be initialized before the actual debug task can be started.

For the actual debugging task the developer may also have to attach several debuggers to the VP. A typical collection of debug tools for the Synopsys Virtual Platform is shown in Figure 1. The Virtual Platform Analyzer (VPA) allows inspecting all details of the hardware platform, while it is possible to connect debuggers to the VPA for debugging the application software and the Linux kernel software.

Setting up the tools and driving the VP into the desired state for debugging the given problem can become a quite time consuming task. Figure