VOSySzator: A flexible embedded RISC-V system virtualizer targeting the cloud

www.virtualopensystems.com

Alvise Rigo, Daniel Raho, Samuele Paone, Timos Ampelikiotis

Introduction

System virtualization in embedded systems is a practice that is commonly used in niche use cases and market segments or in very large markets as the automotive one. In essence, these virtualization solutions tend to be very specific, tailored to a limited set of hardware. But most of all, these solutions result to be expensive, given the high development costs of certified software. There are, however, some products belonging to other market segments that could also benefit from system virtualization like, for instance, network equipment, set-top boxes and kiosks. In such cases, due to the lack of flexible solutions that can be easily tailored to specific use cases, OEMs renounce to the extra benefits of virtualization (like snapshotting, isolation from the host kernel, file system overlays, etc.) in favor of more conventional system designs, believed to be cheaper and easier to maintain.

VOSySzator is meant to democratize virtualization in all markets, by proposing an accessible solution for system designers and integrators to introduce virtualization to all those scenarios that for the aforementioned reasons are keeping a more traditional design. As an answer to this, VOSySzator will make heavy use of device-passthrough, a virtualization technique that allows to expose a device to the guest Operating Systems, allowing it to have full control of the device, with almost native performance.

Core features

The nature of an embedded system where MMIO devices are used poses a considerable challenge when it comes to device pass-through as each device might depend on one or more devices. To have a glimpse of this, it suffices to inspect a flattened device tree (FDT) which is the description of the hardware which is parsed by the kernel. Dependencies like clocks, pins, memory regions, etc. are all encoded in this file, defining a graph of dependencies interconnecting all the devices in the hardware.

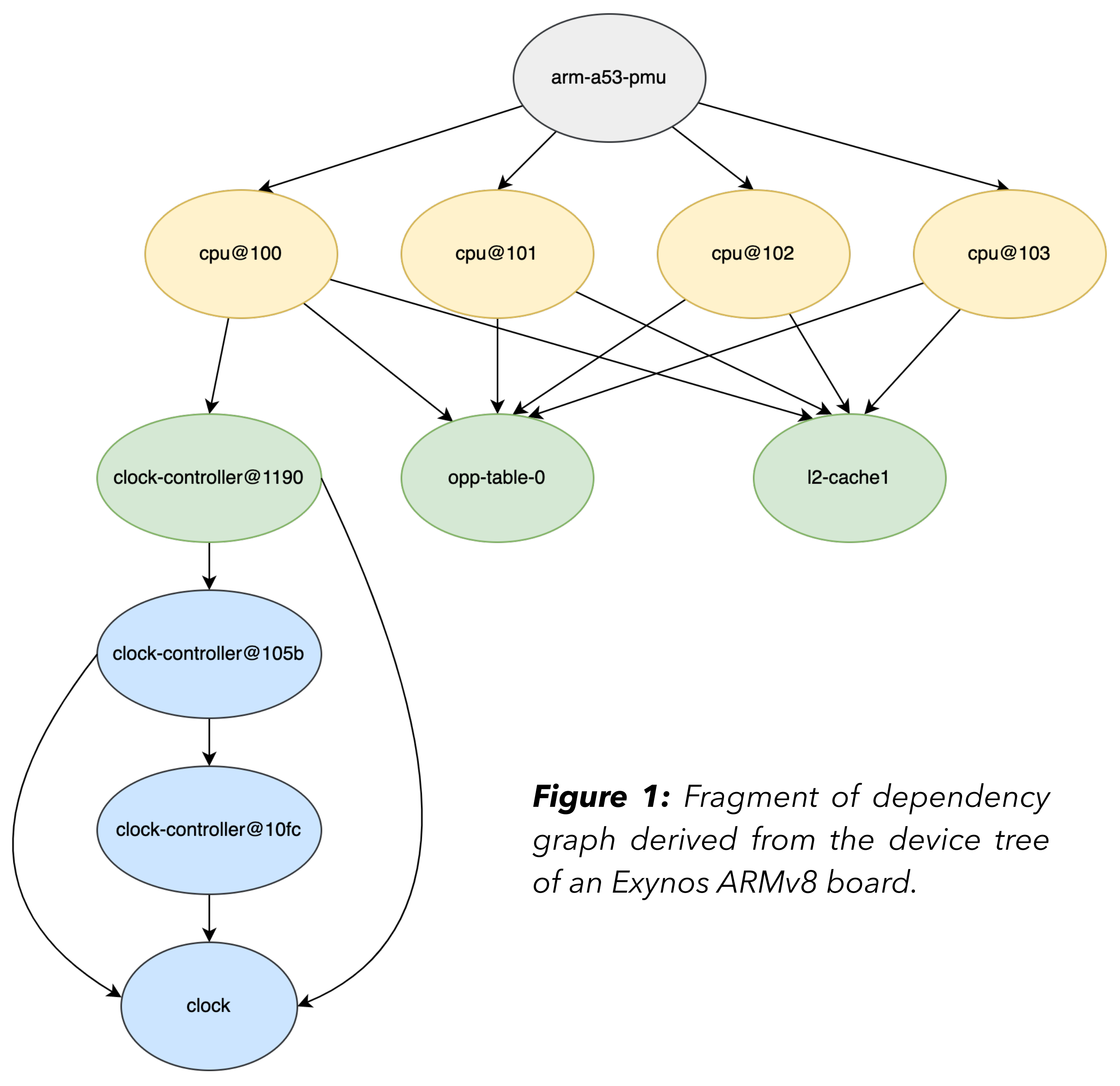

Figure 1: *Fragment of dependency graph derived from the device tree of an Exynos ARMv8 board.*

Except few, non-connected nodes of this graph, all the other nodes exhibit one or more dependencies that must be met also after the device has been attached to the guest virtual machine. VOSySzator proposes a novel solution that analyses such dependencies and tries to satisfy them without the intervention of the user, actuating two different functions: the first one tries to pass-through the dependencies along with the desired device. This is the most straightforward solution which might not work all the time as not all devices can be passed-through.

There are some devices that either because of user's requirement or for technical reasons shall stay attached to the host (and thus bound to the host driver). For example, unbinding a clock controller from the host would harm the stability and functionalities of the whole system. In these cases, the second function is deployed, which creates a dedicated proxy inside the guest OS to remote the functionality of the unmet dependency from within the guest to the host via a vhost-based mechanism as shown in Figure 2.

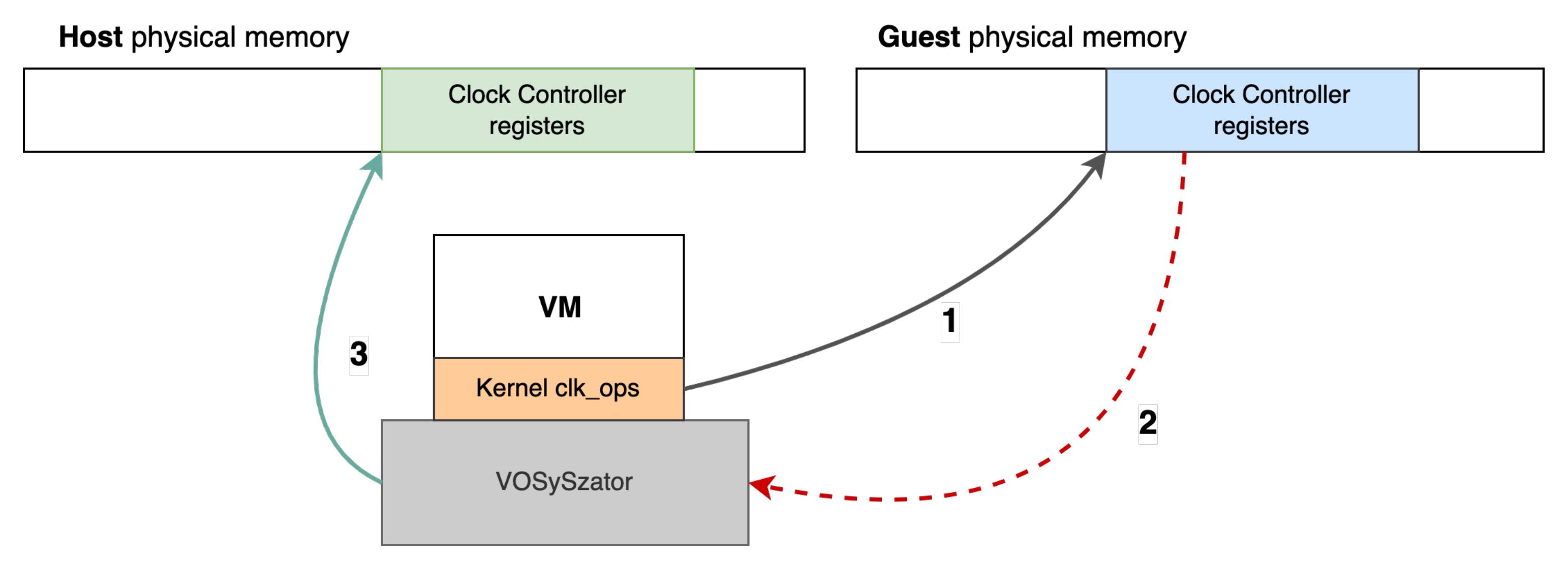

Figure 2: *In this example, the guest accesses to the clock controller are mediated by VOSySzator, which guarantees that the clock operations being called by the guest do not harm the stability of the host system. The guest access (1) targets a region of guest physical memory instead of targeting the real hardware. This region is configured to trap the access (2), letting VOSySzator the duty to handle it. After the needed verification, the access will be replicated to the real hardware (3).*

Benefits

A well-thought, user-friendly and flexible solution for device-passthrough brings benefits that are not exclusively about performance. In fact, the configuration and deployment of virtual machines that can offer the same set of devices as the host system is a decisive factor to succeed in running existing BSPs or system images in the guest system with a few to none modifications.

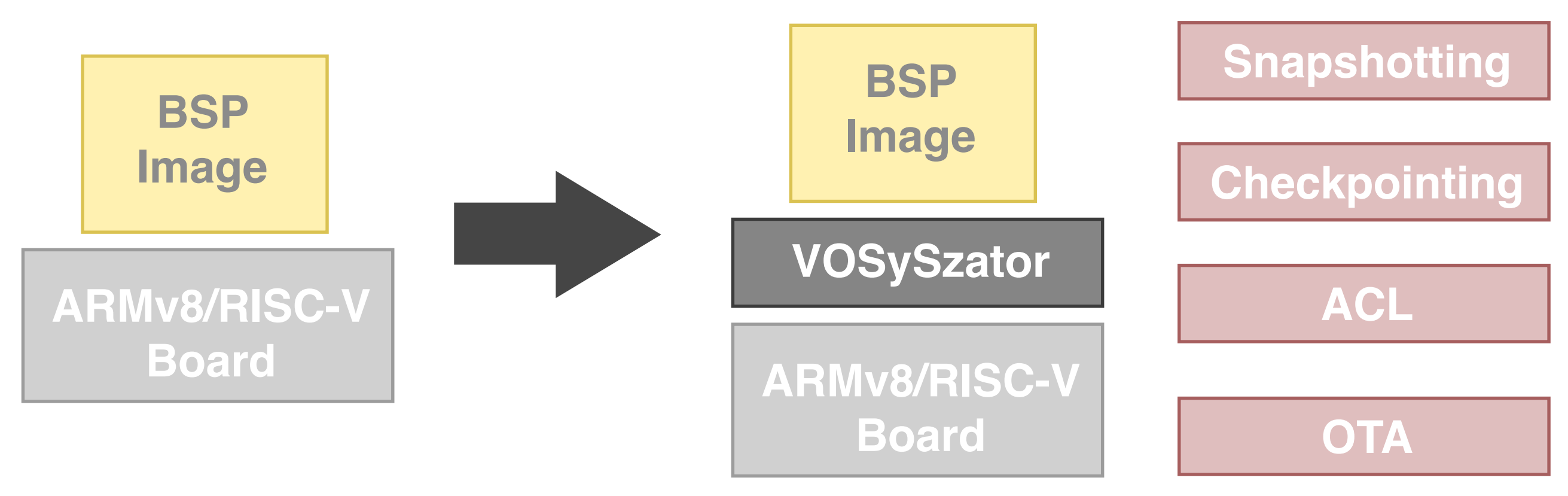

Figure 3: *Running an existing image on top of a system virtualizer brings various benefits, like snapshotting/checkpointing of the virtual machine, a fine-grained control over the device accesses and the possibility to implement flexible OTA solutions that involve minimum down-time for the guest.*

The host system in this case can be the very same hardware that originally runs the image, but also different hardware with the same set of devices. To support this use case, VOSySzator will provide the means for the user or system integrator to define a custom physical layout of the guest memory, which is also a handy feature to overcome some limitations due to the lack of an IOMMU.

The concept of device pass-through is not only meant for embedded scenarios. Also in the cloud, specifically in HPC applications and services, VFIO can be used to expose accelerators into the guest, to grant maximum performance while ensuring the complete isolation of the HPC tasks. In these scenarios, VOSySzator, which aims at making the pass-through experience as integrated and smooth as possible, can help encouraging the deployment of RISC-V-based cloud HPC solutions in those cases where the availability of a few virtual machines equipped with powerful accelerators is more important than many general-purpose guests. At last, VOSySzator's trap-and-replicate mechanism used to proxy guest's device accesses to the real device can be used to share the same device with multiple VMs by implementing a thin emulation layer in VOSySzator.

Conclusions

In embedded systems, virtualization is a powerful concept that gives system designers and integrators the opportunity to explore new solutions to create new BSPs, firmwares and system images that are easier to develop, debug, test and maintain. At the same time, RISC-V is gaining more and more ground in the IoT, smart appliances, home automation, automotive, industrial and, in general, in cyber physical systems and is expected to gain popularity in cloud systems as well. In this perspective, VOSySzator aims at providing the key functionalities to design and implement system software for this new wave of devices, as well as to provide the tools to move existing software from a bare-metal to a virtualized execution, with all the added value of virtualization.

Virtual Open Systems
@VOSySofficial

Funded by the European Union. Views and opinions expressed are, however, those of the authors only and do not necessarily reflect those of the European Union or the HaDEA. Neither the European Union nor the granting authority can be held responsible for them. Project number: 101093062