Modeling Virtual Machines and Interrupts in TLA$^+$ & PlusCal

Valentin Schneider
Arm Limited, UK

The Arm Generic Interrupt Controller (GIC[1]) is a hardware component that manages the interrupts in a system. When there are pending interrupts (e.g. a network card has received a packet and raised an interrupt), it will signal them, which will lead to triggering a CPU exception. The GIC also provides interfaces (registers) to handle these interrupts, or even to generate new ones (software-generated interrupts).

Furthermore, the GIC includes virtualisation extensions. These extensions are used to forward interrupts to Virtual Machines (VMs, or guests), either physical (e.g. system timer) or purely virtual (e.g. virtual peripherals). This allows the hypervisor to control which interrupts are visible to the VMs.

In our case, the hypervisor is KVM (Kernel-based Virtual Machine[2]), a virtualisation solution used within the Linux Kernel.

The use of the virtual extensions is not limited to simply forwarding interrupts. A VM needs to be oblivious to which physical CPU it is running on, yet the GIC interfaces are per-CPU. Should we be running a VM with several virtual CPUs (vCPUs) on a single core system, we would need to save and restore the state of the sole CPU interface depending on which vCPU is being run. The hypervisor is tasked with handling these GIC context switches.

This work thus involves two models:

1. A GIC model written in TLA$^+$, which models interrupt states, interrupt signals, active priorities, etc.
2. A KVM model written in PlusCal, which models the different software context switches (Interrupt handler, hypervisor, VM/Guest) as well as the handling of interrupts and interactions with the GIC by both the hypervisor and the VMs.

To make these interactions possible, the GIC specification and the KVM specification share a communication channel. For instance, to write to a register, the KVM specification will append the name of the register and the data to write to a command queue.

[1] `https://www.cl.cam.ac.uk/research/srg/han/ACS-P35/zynq/arm_gic_architecture_specification.pdf`

[2] `https://en.wikipedia.org/wiki/Kernel-based_Virtual_Machine`

The presence of data on this command queue is an enabling condition of a step in the GIC specification that will consume the command, update the internal GIC state and append a reply to the communication channel.

Finally, both the KVM and GIC specifications are conjoined with a common next-state relation. The resulting specification describes behaviors where KVM and the GIC interact via that communication channel.

Using TLC, the goal is to verify:

(a) Safety properties such as interrupts delivered to the correct guests, recovering from misbehaving guests not signalling the end of interrupt, correct context switching of multiple vCPUs running on a physical CPU, correct migration of a vCPU to a different physical CPU.

(b) Liveness properties such as deadlock freedom, interrupts eventually delivered to a guest.