QEMU CPU virtualization with SystemC and Palladium

- By - Harshit Monish <harshitm@cadence.com>; Raghav Mahajan <raghavm@cadence.com>; Prashant Vardhan Agarwal <agarwalp@cadence.com>

Introduction

Virtual prototyping is today an essential technology for modelling, verification and re-design of full HW/SW platforms. It brings several benefits like, for example, efficient management of design complexity, decoupling of SW development from the availability of the actual HW implementation, and control of prototyping costs. A common aspect in modern virtual platform approaches is the use of an instruction set simulator (ISS) that exploits dynamic binary translation (DBT). This technique has become the de facto standard to guarantee high speed and accuracy of cross-compiled software.

With the emulators available today, it is a big advantage to develop applications for hardware without the physical devices itself. Emulator can provide a virtual platform for quickly software development such as Android Emulator, which uses QEMU to emulate the whole mobile platform. Application developers can develop and debug code on the emulator without the real mobile phones, easily and cost saving.

QEMU is a machine emulator relying on dynamic binary translation of the target CPU application code. Each instruction of the target CPU is translated into a set of micro-operations that are implemented by C functions for the host machine. Such C functions are then compiled to obtain a dynamic code generator which is called, at run time, to generate the corresponding code to be executed on the host machine. QEMU is a fast, portable and dynamic binary translator (DBT) that supports a wide range of processors (X86, ARM, PowerPC, MIPS etc.), it can also emulate the whole platform not only PC but also embedded development board.

This paper proposes a CPU virtualization solution for Cadence Palladium emulation solution which enables the customer to verify the combination of functional SystemC models, C models and RTL models in his design. This enables the customer to replace the CPU and GIC component with QEMU virtualized CPU and GIC in his design without relying on the SystemC Fast models or the RTL models of the CPU or GIC, hence reducing the time to boot Linux on it-as we found a performance gain in this solution. It also reduces the cost of verification as Fast Models come with a good price. In the scope of this paper we have virtualized the A53 ARM processor in QEMU and using this processor component booted Linux on the arm_versatile_pb board.

Proposed Solution

In this paper we put forward the approach to virtualize just the CPU in QEMU which can be integrated with individual TLM models written in SystemC, C models and RTL models of the design.

QEMU

In QEMU the communication between a CPU emulator and the emulated devices is done via registered callback functions for each memory region of the system bus. Then, when the CPU emulator does an access to a memory address, the proper callback function (to the device registered with this address range) is called and the functionality of the device is emulated. QEMU devices respond immediately to devices accesses, and do not account for any processing time in the device or any bus occupancy.

SystemC

SystemC is a general simulation kernel designed for models from cycle-accurate and pin-accurate modeling of hardware behavior all the way up to abstract bandwidth and behavior models. Unlike QEMU, SystemC allows for device models to contain active threads (SC_THREAD) as well as purely event-driven objects (SC_METHOD). This is very similar to the process concept of VHDL. SystemC is based on C++, and all objects in the SystemC simulation are C++ objects.
SystemC is an event-based simulator with potentially multiple active simulation contexts. When simulation begins, the kernel finds the top event on its event queue and executes or resumes the thread or method sensible to that event.

System Integration

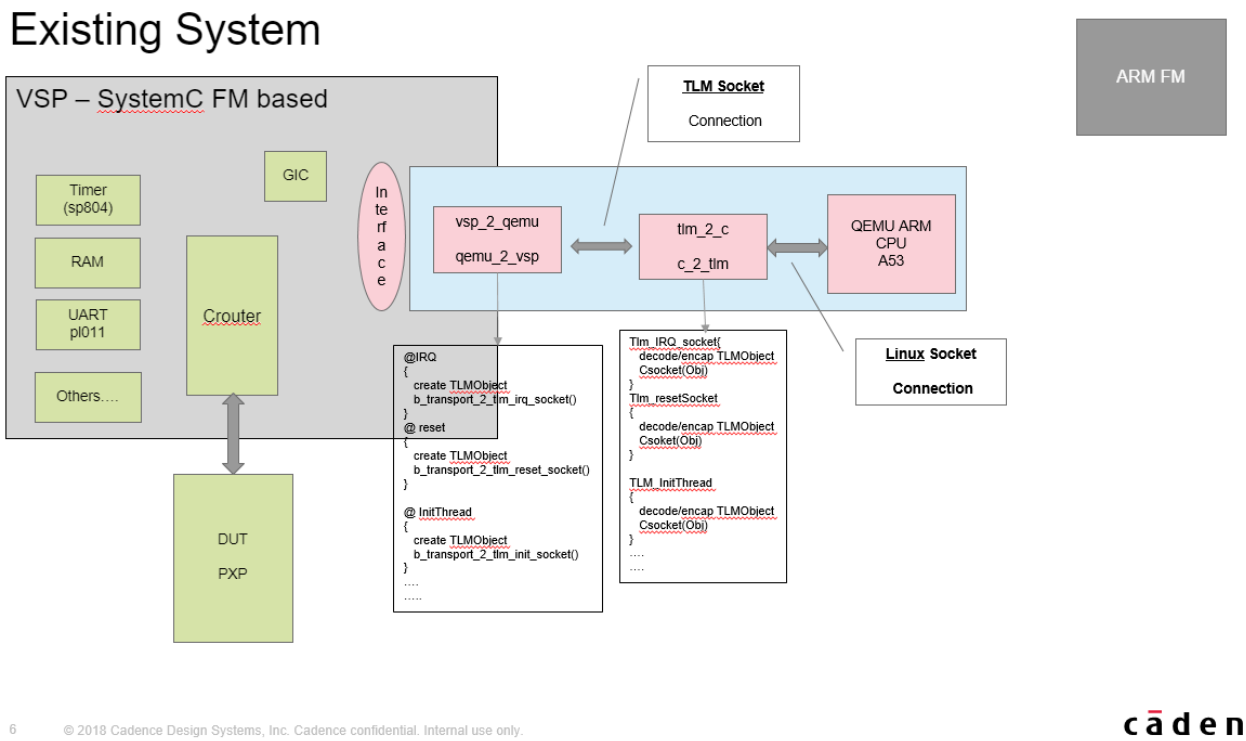

Fig. 1

The existing systems uses the SystemC Fast Models for CPU and GIC provided by the ARM that gets integrated with Crouter (a SystemC component that routes the TLM transaction to a device) and other SystemC peripheral devices are also connected with the Crouter. All the SystemC components uses the TLM methodology for communication. User RTL DUT inside the palladium is connected by creating a

TLM to RTL bridge kind of component which maps the TLM transaction to RTL pin level. This kind of platform involve just the SystemC and RTL integration which depends heavily on ARM Fast Models.

The new proposed architecture replaces the ARM Fast Models of A53 CPU and GIC (interrupt component) with the QEMU virtualized ARM CPU and GIC (generic interrupt controller). These components are then connected to the Crouter in the same fashion as the ARM Fast Models and does the TLM communication across the platform.

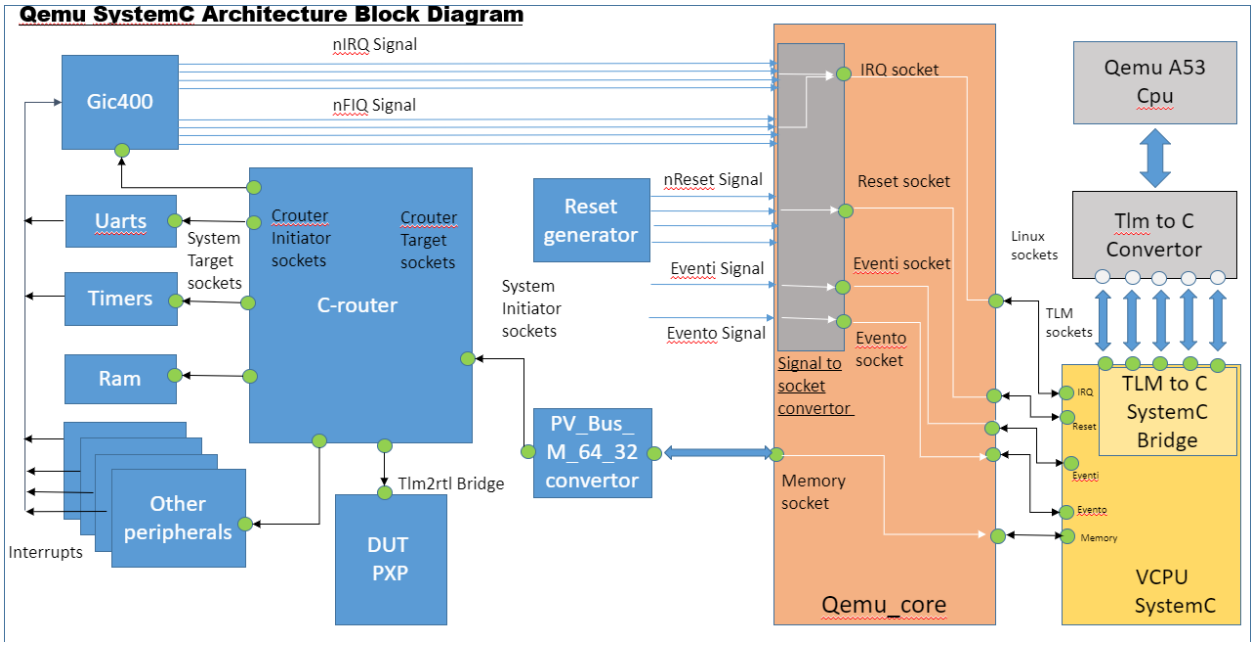

Fig. 2

QEMU V_CPU

Key points:

- QEMU A53 CPU model is extracted and wrapped inside the SystemC wrapper. This acts as a SystemC component and gets integrated into the design.
- A TLM TO C convertor is integrated which converts the TLM transactions to the C function calls for the QEMU CPU and vice versa using Linux sockets.
- GIC is part of the virtualized CPU and the interrupt signals are taken as an input into the V_CPU. The memory calls are directed through the memory sockets.

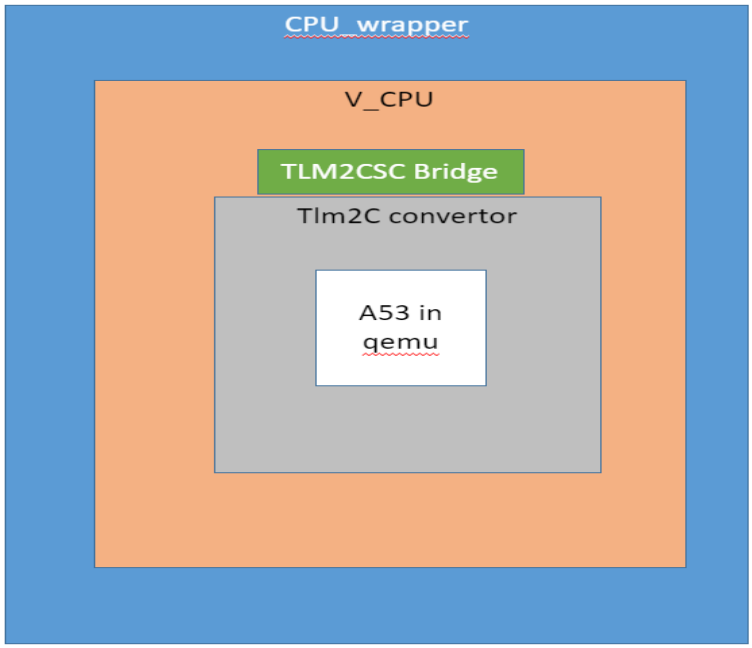

Fig. 3

Experimental results:

The solution works properly, and we have successfully booted Linux image on the V_CPU integrated in arm_versatile_pb board. Also found the performance gain and will be adding actual numbers as part of the paper.

Conclusion/Summary

The paper demonstrates a novel and scalable approach to virtualize the CPU in QEMU and covers various use models of it. This solution will be helpful for the Cadence Customers as it enables the integration of various C, SystemC and RTL models. It also helps in reducing the cost of verification as it replaces the ARM Fast Models with QEMU virtualized models which are open source.

Who benefits from this:

- This solution helps customers in giving the head start of software development even if the RTL models are yet to be completed.
- It also reduces the cost of verification for customers as customer has to bear the cost of ARM Fast Models.

Future Work:

The solution can be further extended to virtualize other ARM CPUs as well. It can also be extended to virtualize other architecture CPUs as well i.e. MIPS, x86, etc. It also can be extended to integrate smart memories and cache coherency solutions.