

Mixed SW/SystemC SoC Emulation Framework

Màrius Montón, Antoni Portero, Marc Moreno[†], Borja Martínez, Jordi Carrabina
CEPHIS

Universitat Autònoma de Barcelona
Bellaterra, Spain

marius.monton@uab.cat, antoni.portero@uab.cat, borja.martinez@uab.cat, jordi.carrabina@uab.cat

[†]marc.morenob@uab.cat

Abstract—Developing HW modules for standard platforms like PCs or embedded devices requires a complete system emulator availability to detect and fix bugs on developed HW, Operating Systems (OS) drivers and applications. This paper presents a set of plug-ins to an open-source CPU emulator that enables mixed simulations between platforms emulators and hardware (HW) modules described in SystemC. In this paper three plug-ins for QEMU are described: one for connecting TLM SystemC modules to any bus QEMU emulates, one for connecting SystemC to PCI bus for PC based platform and one plug-in for connecting SystemC to AMBA bus for ARM platforms. With this framework, it is possible to develop OS drivers at the same time HW is developed and final application tested running in this virtual platform.

I. INTRODUCTION

Current systems are based on a standard CPU with a set of peripherals for interfacing other devices and specific hardware modules for algorithms acceleration. These systems usually run an Operating System (OS) which manages the applications. Then, applications access peripherals using OS drivers, each of them specially written to interface with each peripheral. When developing or using a new HW module in a platform, driver development and application coding is also necessary.

Using custom modules, designers can develop their own hardware modules to fit exactly their applications and specifications. Dedicated HW modules are necessary to obtain systems working correctly. It is also required to write correct drivers for the OS and modify application to use those modules. Furthermore, a large amount of simulation cycles are required starting with atomic test benches of HW modules, and ending with whole system test. Hence, new methodologies are essential to increase engineer productivity and decrease time to market. Also, a set of new tools is required in order to find bugs than can be very hard to find in real systems that only appear when running whole system all together. This paper is divided in five sections: Section II describes previous work in this topic. Section III introduces the simulator we used. Sections IV and V describe different work done with this simulator. And paper finishes showing some results and future work.

II. PREVIOUS WORK

Several proposals of HW-accelerated simulations frameworks based on FPGAs appeared in [1][2][3]. Those frameworks improve SW cycle-accurate simulators running system

into some reconfigurable devices and extracting all information and all desired traces. However, working with real hardware and synthesizable description of HW modules is inefficient in early stages of development. Also, the usage of TLM SystemC to model and simulate entire SoC [4][5], has been proposed. In this approach, the main problem is to get an accurate and realistic CPU model, good enough to allow running real Operating System on it; also, the use of SystemC models is very CPU expensive to simulate entire and complex systems.

A behavioral model is a better approach for HW components of system to test performance and correctness of HW/SW partition used.

Our proposal is to exploit a SW CPU emulator to emulate entire system, avoiding cycle accurate simulation. Afterward, system is plugged-in to a cycle-level simulator only for HW module that is in development stage. With our framework, early tests of correctness of HW/SW partitioning can be achieved using behavioral description of HW modules. Also parallel SW, O.S. drivers and HW development can be done currently by different work teams.

III. QEMU

QEMU is a generic open source processor emulator [6] that can emulate entire system based on supported CPUs: Intel x86 and x86_64, ARM, SPARC, PowerPC and MIPS. QEMU can also emulate entire systems based on these CPUs with most common peripherals used: ARM's Integrator/CP and Versatile/PB boards, Intel based PC, AMD64 based PC, SH4 SHIX board and Power Macintosh. These emulated systems are able to run unmodified standard Operating Systems like GNU/Linux or MS-WindowsXP.

Our main contribution is that we have added a set of plug-ins to QEMU in order to be able to emulate SystemC [7] modules together with QEMU, allowing these SystemC modules run as peripherals of the emulated platform.

We focus our main effort from the set available in two systems platforms: (1) Standard Intel x86 PC Based platform and (2) ARM's Versatile development board, an ARM926EJ-S based platform [8].

For both platforms, two emulation levels are available: TLM and RTL. Using TLM, faster prototype of HW module can be written and plugged into emulation platform to test correct behavior and HW/SW partitioning. Once requirements are

satisfied, specification enhancement at SystemC-RTL level can help designer to develop the real interface.

IV. TLM EMULATION MODE

At TLM emulation level, communications between QEMU and SystemC modules are made using transactions. At this level, write and read operations from CPU to system bus used in emulated platform are sent using a TLM channel to the SystemC module. Basic structure of this emulation mode is shown in Fig. 1.

In this emulation mode, every access to the bus involving the target device, QEMU is freeze and SystemC simulation is started to perform the bus transaction into TLM channel (read or write). Once it is finished, QEMU is resumed, bus access is finished and virtual system simulation continues properly.

This emulation mode allows simulations of the entire system in early stages of development, enabling full system test, correctness of HW/SW partition, developing O.S drivers and ckecking right behavior of module to be synthesized.

This channel can be built simulating the behavior and restrictions of real bus used in order to extract time outcome, accurate enough to test if requirements are met.

Moreover, TLM emulation can be used to extract data traces that are passing through the virtual bus. These data values can later be used to perform some exhaustive and more realistic simulations of the SystemC module that is under development.

Additionally, TLM emulation can be used to develop O.S. drivers and test complete application (SW + driver + HW) all inside the virtual platform.

TLM channel developed is valid for all platforms described, appearing for each case as a standard device connected to the system bus (AHB Slave device for ARM platforms, PCI target device for PCs, etc.). In all cases, TLM transactions are the same, allowing the use of the same SystemC module for testing in all diverse platforms.

V. RTL EMULATION MODE

At RTL emulation level, communication between QEMU and HW module is made using each emulated standard platform i.e. PCI transfer in PC platform, AHB transfers in ARM's platform etc.

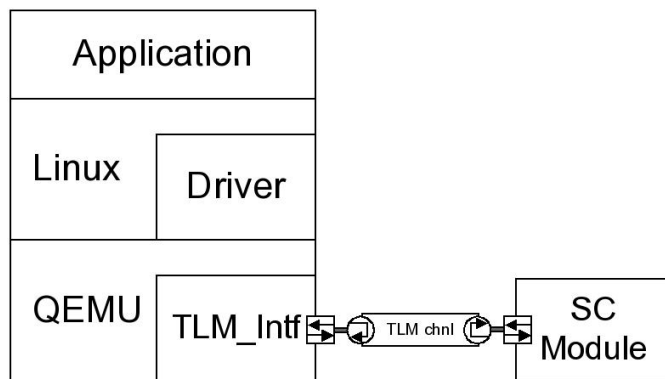


Fig. 1. Diagram of the QEMU-SystemC emulator framework

In this way, once the design is validated into emulated system, it can be synthesized and plugged into the real platform without changing neither hardware nor drivers nor application.

In next subsections the plug-ins done for different platforms are presented:

A. Intel x86 PC Platform

QEMU can emulate a complete PC platform and run any standard OS without changes. Platform peripherals provide external communications such as Ethernet controller, VGA card, Hard Disk controller, keyboard and mouse devices, etc.

We have developed a plug-in to virtual PCI bus on QEMU to enable the connection between SystemC modules and the platform mentioned before. As seen in Fig. 2 (a), we emulated an Altera PCI Core [9] as interface between QEMU and SystemC (PCI_SC). At this emulation level for this platform, we inter connect the virtual PCI device with QEMU emulator platform in the same way Altera Core do. All accesses to this PCI device are captured and re-directed to SystemC module through a SystemC wrapper (PCI_Intf).

Once an access to PCI target device is done by CPU, PCI_Intf starts SystemC simulation: PCI_SC emulates the real Altera PCI core behavior and SystemC module receives same signals set that would receive when it is synthesized and connected to a real Altera PCI core. PCI_Intf code starts SystemC simulation (sc_start()) and, after initialization, runs it for 6 clock cycles for a PCI-33MHz read or write transfer (non-burst). If PCI transfer is not yet finished (e.g. due to a high latency device), PCI_Intf loops for more clock cycles until PCI transfer finishes.

B. ARM Versatile Platform

ARM Versatile Platform (AVP) is an ARM926EJ-S based board especially designed to prototype systems based on AMBA AXI and AHB bus interfaces. This board is based on a large FPGA that implements the bus infrastructure and can be used to prototype custom AMBA peripherals [10].

QEMU emulates this board offering the same peripherals present in AVP board. Furthermore, it can be possible to run GNU/Linux into that emulated platform.

We have developed a plug-in that enables communication between AMBA bus present in this platform and a HW module described in SystemC. These inter communications are transformed into AHB transfers in the SystemC simulator. In

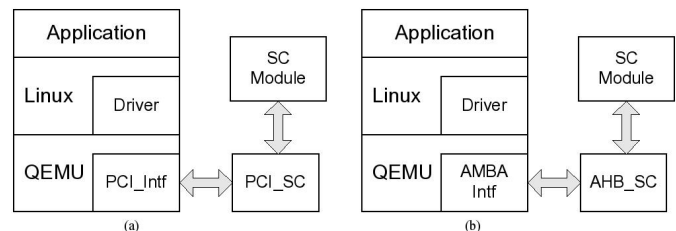


Fig. 2. QEMU-SystemC for PC platforms (a) and QEMU-SystemC for ARM platforms (b)

TABLE I
EXECUTION AND SIMULATION TIME OF MPEG-2 DECODER (TIME IN SECONDS)

	Real Platform	QEMU	Overhead
i386 PC	0.088	0.638	7X
ARM-9	1.06	1.5	1.4X

this way, SystemC module receives input AHB signals in the same way it would be received in a real AHB bus transaction.

Each access to any bus address corresponding to a HW module address is captured. Then, QEMU freezes and AMBA_Intf reproduces signal values of real AHB bus to SystemC AHB slave module. Later, SystemC simulation is started (`sc_start()`) and two cycles later the produced result is taken from Slave Module. Those results are passed back to QEMU in order to complete the access to our SystemC module by emulated platform (Fig. 2 (b)).

VI. EXPERIMENTAL RESULTS

We developed and tested our emulation framework with several applications. Following we describe the obtained results

A. QEMU Overhead

Our first experiment has been devoted to evaluate how efficient is to run a complex application into the emulated platform using QEMU.

We have ran a standard MPEG-2 decoder in a real ARM platform and an equivalent QEMU emulated platform (ARM's Versatile PB926EJ-S). Also, we run the same application on standard PC (Pentium-D@3 GHz) and emulated PC by QEMU. In all cases, QEMU is running on a Pentium-D@3 GHz desktop PC.

Results in Table I show execution time of decoding a 0.24 seconds of 352x240@30fps MPEG-2 stream. As seen, using QEMU is about 7 times slower than the real platform when emulating PC platform, and only about 1.5 times slower when emulating ARM systems. This overhead is much better than simulating the entire system at cycle-accurate level.

Note that to obtain the total time you should add the time boot of Operating System (several tens of seconds, near 2 minutes of real execution time) must be added to total time. This long simulation time make impossible to use cycle-accurate simulator for test of complex systems but it is possible using QEMU.

B. MPEG-2 decoder

Next set of experiments using this framework involve profiling the application and developing a special HW module for the two most costly functions. These functions are `idctcol` and `idctrow` in our MPEG standard decoder, where CPU spends about 20% of computing time. These two functions made Inverse Discrete Cosine Transform (IDCT) by files and rows.

SystemC code for these two functions is integrate into virtual AVP platform as two HW modules both plugged to AHB bus. Also, GNU/Linux driver was written and main

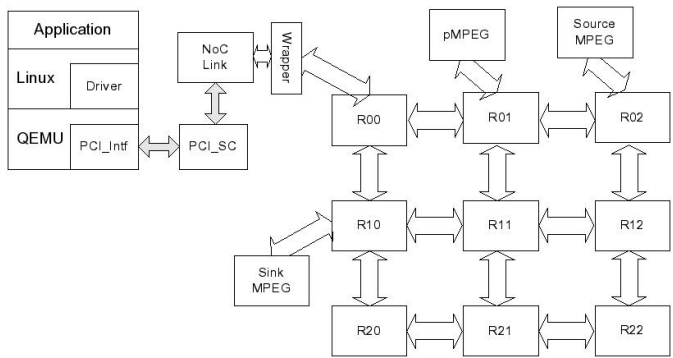


Fig. 3. QEMU-SystemC with MPEG-4 NoC system

MPEG-2 decoder was modified to access to this two functions through especial O.S. devices (`/dev/idctcol` and `/dev/idctrow`).

All this code modifications were done inside QEMU, and it should be the same kind of development needed to use these modules in the real final platform.

Simulation time with QEMU is about 50 seconds running on a Pentium-D@3GHz for decoding 0.3 seconds of 350x240@30FPS MPEG-2 stream. This time involves only execution time for MPEG decoder, before that, QEMU has to boot the O.S and load the driver.

C. NoC MPEG-4 Encoder

In our third experiment, we attached a complex system described in SystemC to the same ARM and Intel platforms as previous experiments but using TLM interface between QEMU and systemC.

In this experiment, we used a MPEG-4 encoder implemented using a Network-on-Chip (NoC) [11] [12]. This NoC is described using SystemC, as also modules of the MPEG-4 codec that form the tiles inside the NoC, as is shown in Fig. 3. In the NoC, there is just a task that is an MPEG4 encoder. The communication between the resource and the router is store-and-forward: all packets that compound an encoded macro block are stored in the router and are not sent until all NxM packets have been received.

SourceMPEG tile is responsible to capture RGB images and transform them to YUV channels. Afterward, they are sent to pMPEG tile to compress them according to MPEG standard. Final compressed image is sent to SinkMPEG tile. The O.S tile is the responsible of control the compression of the different macro blocks to obtain different quality vs. bit-rate images.

In this configuration, entire NoC is available to CPU bus as a single device. Using this device, CPU can control main operations of MPEG-4 codec, receive encoded video stream and obtain traces and debug data from NoC.

We developed O.S. driver for both platforms (PCI in Intel platform and AHB in ARM platform). These drivers allow control of MPEG encoder (QoS), get encoded stream and put video frames to encode. With a custom application and through this driver, we can control all these aspects of the MPEG encoder, and also get the resultant MPEG stream in a file in the hard disk.

Simulation time for this complex system is about 80 seconds to encode one IPBB GOP of QCIF video (176x144) to MPEG-4 stream.

VII. CONCLUSIONS AND FUTURE WORK

A new emulation framework for complex platforms was described in this paper, allowing designers to develop and test HW modules for complex applications and its associated drivers in early stages of development process.

This framework is based on an open source and freely available emulator named QEMU, we have added further functionalities such as a set of plug-ins to allow connection of SystemC modules.

This connection between SystemC modules and SW emulator is made in two descriptions levels (1) using TLM channels or (2) RTL interfaces. The use of TLM channels allows rapid prototyping of entire system and successive refinement steps until achieve a RTL description of HW modules.

Moreover, a large set of platforms and architectures can be simulated with low implementation effort and high-enough emulation speed to boot and run a standard operating system and target applications on it.

In future, we plan to add more functionality to this framework to enable emulation of dynamic reconfiguration of HW modules. With this functionality, our framework will be useful to emulate complex systems with FPGAs that enable run-time reconfiguration, and it will require the development of new concept of drivers and operating systems for this reconfigurable peripherals.

This framework is fully available for download at <http://cephis.uab.es/proj/public/qemu/>

ACKNOWLEDGMENT

Authors thanks Josep Cañadell for his work in early development stages of this development.

M. Montón thanks Fabrice Bellard and all QEMU community for their support, ideas and suggestions.

REFERENCES

- [1] Chen Chang, Kimmo Kuusilinna, Brian Richards, Robert W. Brodersen, "Implementation of BEE: a Real-time Large-scale Hardware Emulation Engine," in *Proceedings of FPGA 2003*, Feb. 2003.
- [2] N. Genko, D. Atienza, G De Micheli, J.M. Mendias, R. Herminda, F. Cathoor, "A Complete Network-On-Chip Emulation Framework," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition DATE'05*.
- [3] Y. Nakamura, K. Hosokawa, I. Kuroda, K. Yoshikawa, and T. Yoshimura, "A fast hardware/software co-verification method for system-on-a-chip by using a C/C++ simulator and FPGA emulator with shared register communication," in *Proceedings of the 41st Annual Design Automation Conference (DAC)*, pages 299-304, 2004.
- [4] Sudeep Pasricha, "Transaction level modelling of SoC with systemc 2.0.," in *Synopsys User Group Conference*, 2002.
- [5] Jae-Gon Lee, Wooseung Yang, Young-Su Kwon, Young-II Kim, Chong-Min Kyung, "Simulation Acceleration of Transaction-Level Models for SoC with RTL sub-blocks," in *Proceedings of the ASP-DAC 2005, Asia and South Pacific*.
- [6] <http://fabrice.bellard.free.fr/qemu/>
- [7] <http://www.systemc.org>
- [8] <http://www.arm.com/products/DevTools/EB.html>
- [9] http://www.altera.com/products/ip/iup/pci/m-alt-pci_mt32.htm
- [10] ARM, "AMBA Specification Rev 2.0", ARM Limited, 1999.
- [11] A. Portero, G. Talavera, M. Montón, B. Martinez, J. Carrabina, "NoC System for MPEG-4 SP using heterogeneous tiles" in *Proceedings of the XXI Conference on Design of Circuits and Integrated Systems. (DCIS'06)*.
- [12] A. Portero; G.Talavera; M. Monton; B. Martinez;M. Moreno; F. Cathoor; J. Carrabina,;" Energy-Aware MPEG-4 Single Profile in HW-SW Multi-Platform Implementation", in *Proceedings of the International IEEE SOC Conference*, pages 13-16, Sept 2006.