

Full-System Architectural Exploration Sandbox

Eriko Nurvitadhi and James C. Hoe

Computer Architecture Laboratory at Carnegie Mellon

{enurvita, jhoe}@ece.cmu.edu

1. Introduction

The hardware realism and execution performance from prototyping using Field-Programmable Gate Arrays (FPGA) help address the key shortcomings of simulation-based computer architecture research. Although FPGA speed and capacity are approaching a pivotal threshold, creating a microarchitecturally-accurate FPGA prototype of a modern microprocessor remains at the limit of feasibility. On the other hand, an architecturally-accurate FPGA prototype to facilitate the examination of architecture design issues (e.g., new protection and memory models) is well within our reach.

We present our plan to develop a fully-functional IA32 platform (Fig. 1) for architectural exploration where the processor and memory controller are implemented using FPGAs (CPU and M.C. FPGAs in Fig. 1). The malleable FPGA implementations of the processor and memory controllers will allow us to study architectural design changes that are impossible to emulate using commercial off-the-shelf (COTS) components. At the same time, this sandbox will deliver the necessary speedup, over software-simulation, to test new architectural concepts against real OS and applications.

Below, we first give an overview of the planned exploration sandbox. Second, we highlight our current effort to develop an executable IA32 architectural model in FPGA for this sandbox.

2. Full-System Architectural Sandbox

As depicted in Fig. 1, the sandbox system consists of a COTS PC host and a custom processor module (on the host PC's PCI bus). The custom processor module makes use of the host PC's DRAM and peripheral devices (e.g., disk, video, and network) as its own. (The host PC's CPU can be used for bootstrapping and other out-of-band functions, but it does not otherwise participate in the sandbox system's computation.) The custom processing module consists of two FPGAs that implement the IA32 processor and the memory controller. By implementing the processor and memory controller from scratch, we are not bounded by the capabilities and the interfaces of COTS processors and chipsets.

During operation, stock OS and application software execute natively on the FPGA IA32 processor. The FPGA memory controller helps to complete the illusion that the FPGA IA32 processor is the primary processor in the system. This involves remapping bus addresses and redirecting I/O interrupts. (I/O devices would still generate interrupts normally to the host processor, which in turn will inform the FPGA memory controller.)

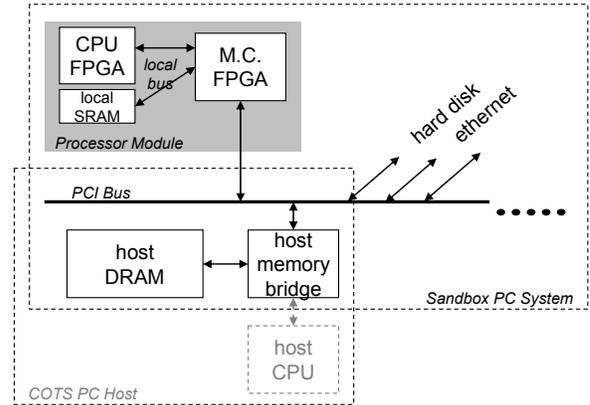


Fig. 1. IA32 Architecture Exploration Sandbox

3. IA32 Processor in FPGA

Since the FPGA IA32 processor only needs to be architecturally-accurate, the implementation complexity and effort are greatly reduced---imagine the difference between a gate-level P4 design versus a synthesized 8086. We are free to choose a straightforward microarchitecture and make use of high-level description/synthesis technologies. This simplistic implementation strategy is central to the usability and reusability of the sandbox infrastructure. Despite compromises in efficiency and optimality, the IA32 architectural model would fit easily in the current generation of FPGAs and can deliver many tens of million-instruction-per-second (MIPS).

Perhaps the greatest obstacle in creating an IA32 processor is in correctly capturing the entire IA32 ISA with all of its subtleties and vagueness. To circumvent this issue, we will assume Bochs¹ source code as the IA32 definition. Bochs is an open-source full-system IA-32 emulator written in C++. Bochs is sufficiently complete and stable to boot a number of Linux and Microsoft operating systems.

In addition to being unambiguous, the structure and organization in Bochs source code further assist our effort. For example, we can derive straightforwardly from Bochs source code the 927 distinct IA32 instruction behaviors. Bochs' instruction lookup tables also shed light on the nuances of IA32 instruction decoding. Finally, we can execute Bochs as the reference behavior for debugging and bringing up of our system.

¹ <http://bochs.sourceforge.net>