Modelling HW/SW Co-Designed Processors

José Cano[*,1], Aleksandar Brankovic[*,1], Rakesh Kumar[*,1], Darko Zivanovic[*,1], Demos Pavlou[†,2], Kyriakos Stavrou[†,2], Enric Gibert[†,2], Alejandro Martinez[†,2], Gem Dot[*,1], Fernando Latorre[†,2], Alex Barcelo[*,1], Antonio Gonzalez[*,†,2]

[*] *Universitat Politècnica de Catalunya, Jordi Girona 1-3, 08034, Barcelona, Spain*

[†] *Intel Barcelona Research Center (IBRC), Jordi Girona 29, 08034 Barcelona, Spain*

ABSTRACT

This paper presents DARCO, an extensible platform for modelling HW/SW co-designed processors with different guest and host ISAs. Its Emulation Software Layer (ESL) provides staged compilation, which translates and optimizes x86 binaries to run on a PowerPC processor. In addition to the functional models, DARCO provides timing simulators and a powerful debugging toolchain. DARCO has a functional emulation speed of 8 million x86 instructions per second.

KEYWORDS: Co-designed processors; dynamic binary translation

1 Introduction

Hardware/Software co-designed processors gained a great momentum during last years. These architectures have important advantages over traditional hardware-only systems like the exploitation of dynamic information, the ability to provide Front-Ends of different ISAs and the segment-specific optimizations. Recently, academia and industry focused on software layers running on top of off-the-shelf processors with the Java runtime and managed systems like the Microsoft .NET being the main examples. Efforts based on HW/SW co-design however have been sporadic and the field has not yet reached a critical mass.

This paper presents DARCO, an effort of multiple man-years that led to a powerful infrastructure for research on the HW/SW co-designed paradigm. To enable a broader research spectrum DARCO has a different host and guest ISA. In particular, it provides a full-system x86 guest Front-End that is translated, optimized and executed on a Power PC

[1]E-mail: {jcano, abrakovic, rkumar, dzivanov, abarcelo}@ac.upc.edu, gem.dot@est.fib.upc.edu

[2]E-mail: {demos.pavlou, kyriakos.stavrou, enric.gibert, alejandro.martinez, fernando.latorre, antonio.gonzalez}@intel.com

(PPC) processor. The key components of DARCO are the x86 and PPC functional models, the Emulation Software Layer (ESL), the timing simulators and the accompanying debugging and monitoring tools. Except for the functional models for which we used heavily modified versions of QEMU [Bel05], all other components are in-house developments.

2 DARCO Infrastructure

DARCO is an infrastructure for research on co-designed virtual machines. It emulates a co-designed processor which is executing x86 applications on a PPC processor through dynamic binary translation and optimization. DARCO provides an Emulation Software Layer (ESL) which implements the layer of abstraction between the target x86 ISA and the host PPC hardware. The reasoning behind selecting x86 as the target ISA is its wide usage. As for the selection of PPC as the host ISA, it is also based on its wide usage and its support for vector instructions on which we can map several of the x86 SSE instructions.

DARCO consists of four main components which interact as depicted by Figure 1. These are: the x86 component, the PPC component, the Timing simulator and the Controller.

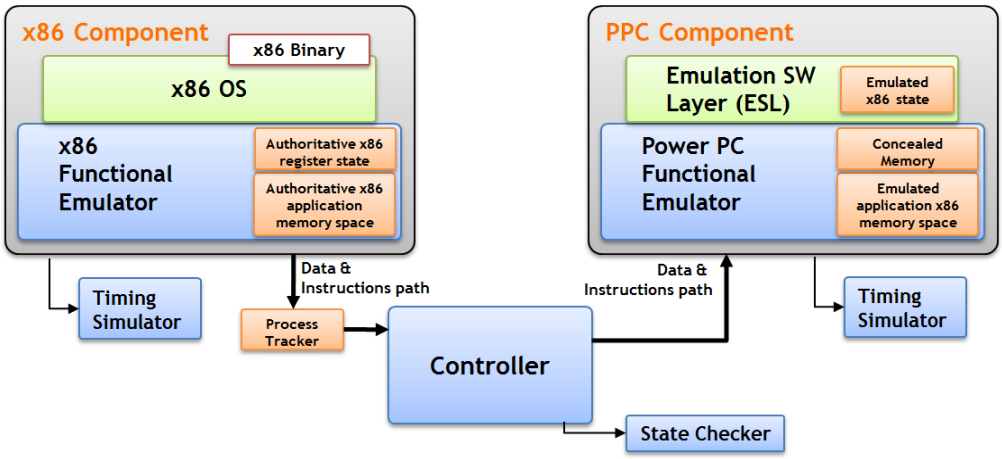

Figure 1: DARCO main components.

The x86 component provides an x86 full-system functional emulator on top of which an unmodified operating system is executing. This component keeps the correct architectural state and is the only component that interacts with the Operating System.

The PPC component provides the processor functional model for DARCO. The PPC functional emulator is executing the ESL which translates and optimizes the x86 instruction stream to PPC instructions. The PPC component has been updated to support additional instructions used by the ESL.

The Timing simulator models a parameterized PPC in-order core. It receives the dynamic instruction stream from the PPC component and provides detailed execution statistics. The use of the Timing simulator is optional and does not affect the functionality of the rest of the components.

The Controller is the main interface of DARCO with the user. It provides full control over the execution of the application as well as debugging utilities. The main task of the Controller is the synchronization of the execution of the other components and the resolution of the various requests from the PPC component.

2.1 Execution Flow

The execution flow of an application passes through three distinct phases; initialization, execution and synchronization.

During the initialization phase, the controller first starts the PPC component which in turn, initiates the execution of the ESL. The PPC component then remains idle until the Controller sends the x86 register state of the application to be executed to it. The x86 component, when launched, initiates the execution of the application defined by the user. The initialization phase is completed when the Controller sends this state to the PPC component. At this point in time, the x86 register state is the same in both components.

During the execution phase, the ESL begins by executing code from the initial *instrucion pointer* it received during the initialization phase. All changes made to both the x86 register state from the emulation of the x86 instructions and the x86 application are stored in the memory space of the ESL. While the x86 application is making forward progress under the ESL, the x86 component remains idle and its memory state untouched.

The synchronization phase is initiated by the PPC component when any of the following three events occur during the execution phase; (1) data request, (2) system call or (3) end of application. After every synchronization phase, the x86 application state, register and memory, is identical between the x86 component and the ESL. Otherwise the system complains and execution is aborted. This is also a useful technique to debug DARCO.

2.2 Emulation Software Layer (ESL)

The ESL is the software layer that executes on-top of the PPC processor. It is responsible for translating the target x86 code to the host PPC ISA. In a nutshell, the ESL has four different execution modes; interpretation mode (IM), basic block translation mode (BBM), superblock and optimization mode (SBM) and code cache execution mode (CCExec).

The ESL starts by interpreting the x86 instructions. When a basic block (BB) executes more times than a predefined threshold the ESL switches to BBM for this BB which is translated and stored in the code cache. The subsequent execution of this BB is done in CCExec mode and profiling information is gathered regarding the direction of the branch and its target. When a BB reaches another repetition threshold, it triggers SBM. During this mode, the control flow profiling information that was collected during the CCExec mode is used by the translator in order to create a superblock (SB) with starting point the BB that triggered SBM. The SB passes through several optimizations and is stored in the code cache. Subsequent executions of the SB are done in CCExec mode.

3 Experimental Results

In this section we present information about the speed of the infrastructure using SPEC CPU 2006 benchmarks (some performance characteristics can be obtained in [PBK+11]). The results regard the execution of the first 200 billion instructions. We measure the speed in millions of emulated x86 instructions per second (Figure 2). This shows the rate of how many x86 instructions pass throughout the execution flow of DARCO which includes all the components of the infrastructure. On average the execution rate of DARCO is approximately 2 million x86 instructions per second.

A parameter that has a definitive impact on the execution speed is the host processor. The results reported are on a cluster where only one core is devoted per execution task. All the three processes of DARCO (x86/PPC component and controller) have to share this one

only core which de-grades the performance significantly. On typical dual core machines, the execution rate averages on 8 million instructions per second.

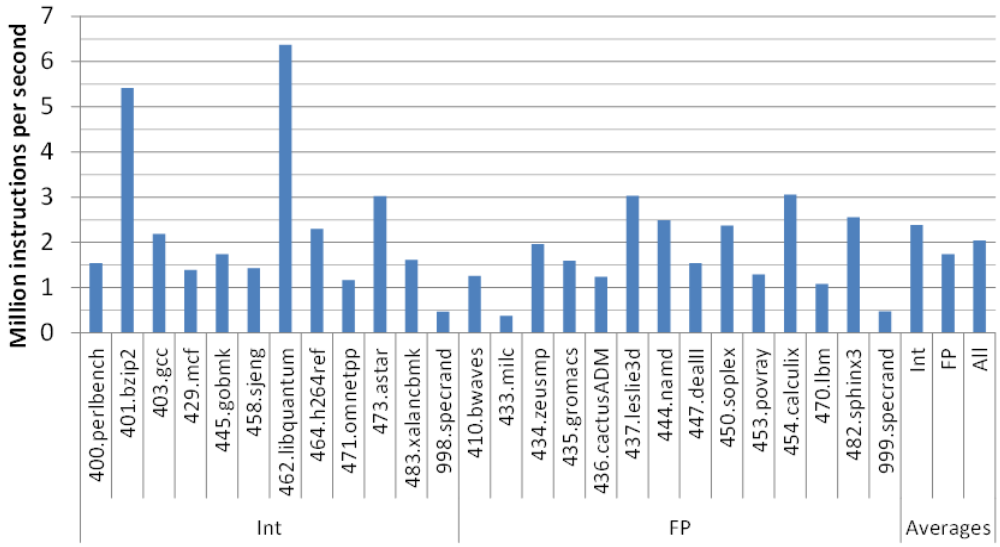

Figure 2: DARCO execution speed.

4 Conclusions

This paper presented DARCO, a complete infrastructure that enables research on HW/SW co-designed processors. DARCO is not an early version of an envisioned infrastructure but a mature ready-to-use and debugged tool. DARCO interprets, translates and dynamically optimizes x86 binaries in PPC instructions which execute on top of a functional PPC emulator. Its Emulation Software Layer includes an interpreter, a translator, a scheduler, a register allocator and a staged optimizer. The other key components are the controller that the user interacts with and the timing simulators.

Acknowledgment

This work is partially supported by the Generalitat de Catalunya under grant 2009SGR1250, the Spanish Ministry of Education and Science under contract TIN2010-18368, and Intel Corporation.

References

[Bel05] Fabrice Bellard. Qemu, a fast and portable dynamic translator. In *Proceedings of the annual conference on USENIX Annual Technical Conference*, ATEC '05, 2005.

[PBK+11] D. Pavlou, A. Brankovic, R. Kumar, M. Gregori, K. Stavrou, E. Gibert, and A. Gonzalez. Darco: Infrastructure for research on hw/sw co-designed virtual machines. In *Proceedings of the 4th Workshop on Architectural and Microarchitectural Support for Binary Translation (AMAS-BT'11), held in conjuction with the 38th International Symposium on Computer Architecture (ISCA-38)*, June 2011.