ArcSim Documentation

Flexible Ultra-High Speed Instruction Set Simulator

ArcSim Documentation
The University of Edinburgh
2011

Version:	2011.1
Date:	March 7, 2011
Status:	draft
Confidentiality:	confidential
Reference	ArcSim-sys-doc

Table of Contents

Chapter 1

Introduction

The ArcSim simulator is a highly configurable ultra-high speed Instruction Set Simulator (ISS). Architectural features such as register file size, instruction set extensions, the set of branch conditions, the auxiliary register set, as well as memory mapped Io and closely coupled memory (Ccm) extensions can be specified via a set of well defined Apis and configuration settings. Furthermore, microarchitectural features such as pipeline depth, per instruction execution latencies, cache size and associativity, cache block replacement policies, memory subsystem layout, branch prediction strategies, as well as bus and memory access latencies are fully configurable.

Overall the simulator provides the following simulation modes:

- *Co-simulation* mode working in lock-step with standard hardware simulation tools used for hardware and performance verification.
- Highly-optimised *interpretive* simulation mode.
- *High-speed* Dbt simulation mode capable of simulating an embedded system at speeds approaching or even exceeding that of a silicon Asip whilst faithfully modelling the processor's architectural state.
- *High-speed* target microarchitecture adaptable *cycle-accurate* simulation mode modelling the processor pipeline, caches, and memories.
- A *profiling* simulation mode that is orthogonal to the above modes delivering additional statistics such as dynamic instruction frequencies, detailed per register access statistics, per instruction latency distributions, detailed cache statistics, executed delay slot instructions, as well as various branch predictor statistics.

Chapter 2

Software Architecture

2.1 Overview

ArcSim is a target adaptable Iss providing full support of the ARCompact™ Isa. It is a full-system simulator, implementing the processor, its memory sub-system (including Mmu, memory mapped Io devices, closely coupled memories, etc.), and sufficient interrupt-driven peripherals to simulate the boot-up and interactive operation of a complete Linux-based system.

ArcSim implements state-of-the-art just-in-time (Jit) dynamic binary translation (Dbt) techniques, combining interpretive and compiled simulation techniques in order to maintain high speed, observability and flexibility.

2.1.1 Dynamic Binary Translation

Efficient Dbt heavily relies on Just-in-Time (Jit) compilation for the translation of target machine instructions to host machine instructions. Although Jit compiled code generally runs much faster than interpreted code, Jit compilation incurs an additional overhead. For this reason, only the most frequently executed code regions are translated to native code whereas less frequently executed code is still interpreted. Using a single-threaded execution model, the interpreter pauses until the Jit compiler has translated its assigned code block and the generated native code is executed directly. But program execution does not need to be paused to permit compilation, as a Jit compiler can operate in a separate thread while the program executes concurrently. This *decoupled* or *asynchronous* execution of the Jit

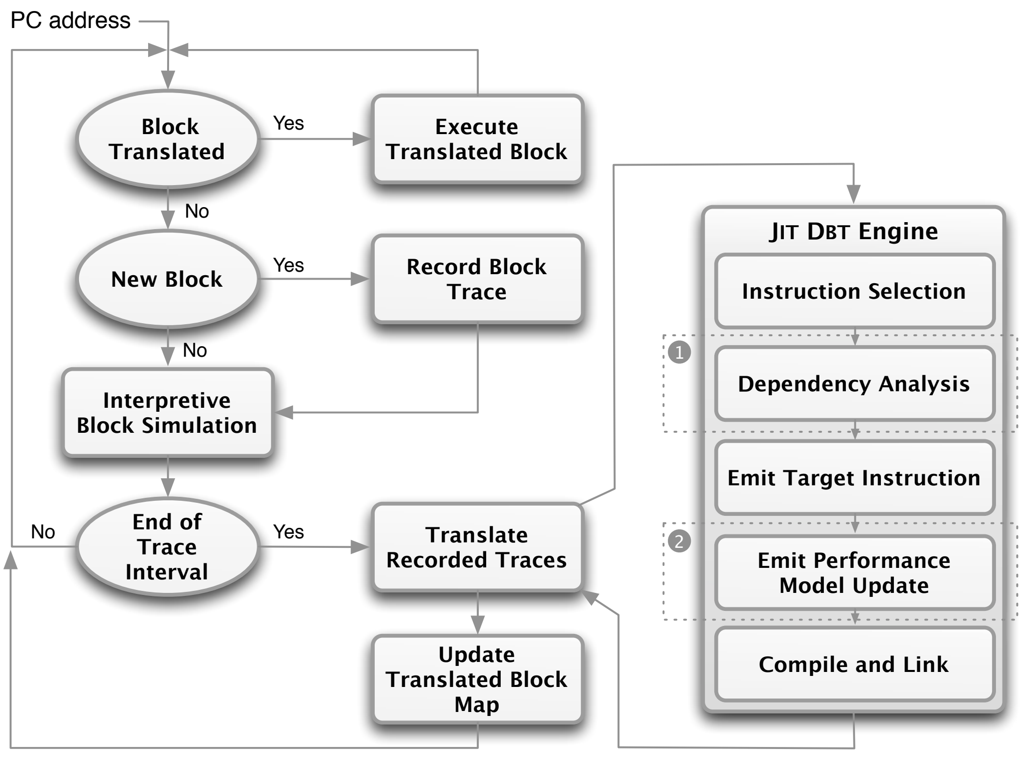

Figure 2.1: JIT Dynamic Binary Translation Flow.

compiler increases complexity of the DBT, but is very effective in hiding the compilation latency – especially if the JIT compiler can run on a separate processor.

ARCSIM implements state-of-the-art JIT dynamic binary translation capable of effectively reducing dynamic compilation overhead, yielding execution speedups by doing *parallel* JIT compilation, exploiting the broad proliferation of multi-core processors. The key idea is to detect *independent*, large translation units in execution traces and to *farm out* work to *multiple, concurrent* JIT compilation workers. To ensure that the latest and most frequently executed code traces are compiled first, we apply a priority queue based dynamic work scheduling strategy where the most recent, hottest traces are given highest priority.

2.2 Simulation Modes

2.2.1 Architectural Simulation

2.2.2 Microarchitectural Simulation

2.2.3 Co-Simulation

2.2.4 Profiling Simulation

2.2.5 Dynamic Binary Translation

Chapter 3

Using ArcSim

3.1 Overview

3.2 Configuration

3.3 Running ArcSim

Running a Simulation

```
$ arcsim -e binary.x
```

Chapter 4

Integrating ArcSim

4.1 Overview

4.2 Application Programming Interface

List of Figures