FAKULTÄT FÜR INFORMATIK

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis

Host Compiled Simulation for Timing and Power Estimation

Gaurav Kukreja

FAKULTÄT FÜR INFORMATIK

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis

Host Compiled Simulation for Timing and Power Estimation

Author:	Gaurav Kukreja
Supervisor:	Prof. Michael Gerndt
Advisor:	Dr. Josef Weidendorfer
Advisor:	Mr. Bo Wang
Submission Date:	15^{th} October, 2014

I confirm that this Master's Thesis is my own work and I have documented all sources and materials used.

Munich, 15th October, 2014 Gaurav Kukreja

Abstract

Simulation is a useful technique for Hardware Software Co-development. It is performed at various levels of abstraction to serve different purposes. Instruction Set Simulation is the lowest level of abstraction where the processor pipeline is simulated in detail, to allow hardware developers to test their modifications and evaluate the impact on performance. At higher levels of abstraction, simulation provides developers with a tangible environment for early software development. The focus of this project is on simulation for performance estimation, namely, estimation of time and power consumed in running a benchmark application on a target processor.

While Instruction Set Simulators are known to be highly accurate, they are difficult to develop and slow to execute because of the level of detail they address. Host Compiled Simulation is a technique to accelerate performance estimation with negligible impact on accuracy. The idea is to instrument[1] the source code, by taking into consideration the behaviour of the target processor. The instrumented source code is compiled and run on the Host Machine. The technique relies on the assumption that performance of each basic block[2] in the binary code can be accurately estimated on a certain processor by emulating the pipeline. Other aspects that affect performance, like resources spent in memory access can be accounted for, and a fairly accurate estimate of the time and power consumed can be estimated.

In this project, a tool to perform Host Compiled Simulation was developed. This thesis discusses the state-of-art in simulation. It explains the approach used to develop this tool. The results showing accuracy of estimations from this approach are presented.

[1]Instrumentation is a technique to modify the source code of an application in order to collect statistics at run-time. This may be used to measure performance of the application, or diagnose errors.

[2]A basic block in a program is a series of instructions which are executed sequentially. The basic block does not contain branch instructions.

Contents

Abstract **iii**

1 Introduction **1**
- 1.1 Simulation : State of art 1
  - 1.1.1 Abstraction Levels 1
- 1.2 Host Compiled Simulation 2

List of Figures **4**

List of Tables **5**

1 Introduction

1.1 Simulation : State of art

Simulation is widely used in Hardware Software Co-development. It allows Hardware Developers to analyze the performance impacts of design decisions at early stage of hardware development. For software developers it provides a tangible environment for early software development.

1.1.1 Abstraction Levels

Simulation can be performed at various levels of abstraction, to serve the different purposes.

Instruction Set Simulation

Hardware developers use Instruction Set Simulator (ISS) to simulate a target processor at the lowest level of abstraction. In ISS, the processor micro-architecture is simulated, taking into consideration details of how each instruction is executed. Accuracy is the focus here, as the simulation is mainly used for analyzing impact of hardware design decisions.

Each stage of processor pipeline is simulated and data and control hazards between instructions are taken into account. Other building blocks like Cache Hierarchy and Branch Prediction units are also simulated, to provide a cycle-accurate estimation of performance.

ISS is difficult to implement and time-consuming to execute because of the details that are addressed. It is not suitable for simulation long running software scenarios. An application that takes hours to execute on the target hardware, may take days to simulate using ISS.

Functional Simulation

Functional Simulation is used to simulate the target processor at a higher level of abstraction. It is mainly used by early stage software developers. It provides them with an opportunity to start developing applications that take advantage of the features provided by the hardware, before it has been fabricated. Performance estimation is not the focus of Functional Simulation.

Simulation functionality in GDB uses this approach. The simulator maintains the state of each register. Each instruction in the binary code is decoded and the state of the registers is modified according to the instruction. However effects due to execution pipeline stages, and caching are ignored.

This approach cannot provide users with accurate estimation of performance.

Sampling based approach

The focus in this approach is to estimate performance. To speed up the execution, the benchmark application is mostly simulated using Functional Simulation, and certain samples are simulated in detail using Instruction Set Simulators approach. The sample size is chosen by the user, based on understanding about the benchmark application. The results gathered from the samples are interpolated to generate performance estimates.

The approach still uses Instruction Set Simulators that are difficult to develop. Accuracy of results depend on the samples chosen. The statistical approach of interpolating results will introduce inaccuracy to the estimates.

1.2 Host Compiled Simulation

Host Compiled Simulation is a technique that improves upon the state-of-art by adhering to following objectives.

- Easier to develop.
- Faster to execute.
- Higher accuracy of estimation.

This technique is based on instrumentation[1] approach. The benchmark application is cross-compiled to generate target binary. The binary is made of basic blocks. In a loop, the same basic block is repeatedly executed. We estimate the time spent in executing each basic block on the processor pipeline, and annotate the matching basic block in source code to accumulate the total time spent when a sequence of basic blocks are executed.

We also instrument the code for c

[1]Instrumentation is a technique to modify the source code of an application in order to collect statistics at run-time. This may be used to measure performance of the application, or diagnose errors.

List of Figures

List of Tables