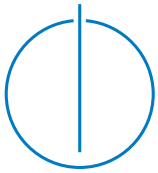
FAKULTÄT FÜR INFORMATIK

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis

Host Compiled Simulation for Timing and Power Estimation

Gaurav Kukreja



FAKULTÄT FÜR INFORMATIK

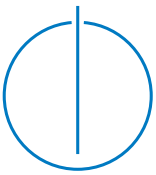
TECHNISCHE UNIVERSITÄT MÜNCHEN

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, 15^{th} 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

1 Introduction

1.1 Simulation

Simulation is the technique to imitate the behaviour of a system. It is generally used when developing or working with the real system is expensive or difficult.

Simulation is widely used in Hardware Software Co-development. The behaviour of a processor is simulated. It allows Hardware Developers to analyse and validate the performance impacts of design decisions at early stage of hardware development. This allows them to save the crucial effort and cost involved in fabricating hardware.

The widely used approach in this area, is called Instruction Set Simulator (ISS). In ISS the processor micro-architecture is simulated in great detail. Each stage of processor pipeline is simulated along with other building blocks of the processor like Caching and Branch Prediction Units. This approach provides cycle accurate estimates of performance. However, ISS is difficult to develop and slow to execute because of the amount of details that are simulated. ISS is not suitable for simulation of long running software scenarios.

Another approach is called Functional Simulation, which is done at a higher level of abstraction. The micro-architecture is not simulated, and effects of caching are ignored. Each instruction is decoded, and the state of registers is modified according to the instruction. This approach is used by early stage software developers, but it is not suitable for analysing performance of the target system.

The focus in this research is to enable fast simulation of single core, embedded processors to provide accurate estimation of time and power spent in executing benchmark applications.

1.2 Related Work

Considerable work has been done in this area of research. The techniques developed can be roughly divided into two approaches.

1.2.1 Sampling Based Approach

Sampling is an approach used in statistical analysis. Small, yet representative samples are chosen from a vast amount of data. These samples are analysed in detail, and the results are interpolated to gather information about the entire data set.

In Sampling Based Approach, certain samples of the execution trace are simulated in detail using Instruction Set Simulation. The rest of the execution is carried out using Functional Simulation at a higher level of abstraction. This leads to a speed up in simulation.

Research in this area mainly focuses on developing procedures to select the samples.

1.2.2 Host Compiled Simulation

Host Compiled Simulation is based on approach of Source Code Instrumentation (SCI). SCI is the process of modifying source code to collect performance statistics and generate trace information during run-time.

In Host Compiled Simulation, the Source Code is instrumented at the basic block granularity to accumulate timing information.

To estimate the performance of an application, we must take into consideration the effects due to following optimization techniques used in modern processors.

- Processor Pipelining
- Branch Prediction
- Memory Caching and Prefetching

The instrumentation enables simulation of these

1.3 Focus

1.3.1 Simple Example

1.4 Thesis Outline

2 Background

3 Host Compiled Simulation

Host Compiled Simulation is the current focus areas of research in the space of simulation. It is popular because it is simple to understand and develop compared to other approaches, and can provide highly accurate estimates of performance.

Host Compiled Simulation (HCS) is based on the approach of Source Code Instrumentation. The source code of the benchmark application is instrumented to collect timing information. The instrumented code is compiled for and run on the Host Machine, hence the name Host Compiled Simulation.

3.1 The Approach

In this research a tool has been developed to automatically instrument the source code of a benchmark application. For accurate instrumentation, the tool analyses the source code along with the cross-compiled binary code generated by the compiler for the target processor.

When a program is executed on a processor, most of the time is spent in executing instructions on the processor pipeline, and fetching data from the memory.

The timing information for time spent in executing instructions in pipeline is annotated at the Basic Block granularity. A Basic Block is a set of instruction which are always executed in a sequence. Each basic block in the binary code is considered. It is assumed that all the data used by the instructions in the basic block is available in L1 Cache, and time spent in fetching data from memory is ignored for now. Each instruction in the basic block simulated to execute on the processor pipeline, and the number of cycles spent is calculated. Cycles spent in pipeline stalls due to interlocking (Control or Data Dependency) among instructions are accounted for. This information is then back-annotated to the corresponding basic block in the source code. Whenever the basic block is executed, the number of cycles spent are accumulated in a global variable.

Note that at the start of each basic block, it is assumed that the processor pipeline is empty, which is not really the case when Branch Prediction Unit correctly predicts the jump from previous Basic Block to current Basic Block. To accommodate for this, Branch Prediction Unit has been simulated. At the start of each branch, an annotation is added which provides the start address of the branch to the Branch Prediction Emulation Unit

which emulates the logic implemented in the target processor to predict branches. If the branch was predicted correctly, a certain number of cycles are subtracted from the total time to remove the penalty which was already added.

To annotate the timing information for memory access, Cache Hierarchy must be simulated. To do this, each load/store instruction in the binary is identified and mapped to an instruction in the source code. To accurately simulate memory access, we need to reconstruct the memory address for the corresponding variable being fetched on the target system. We then simulate the access to this address in the cache. The cache simulator checks whether the data is present in one of the caches, or should be fetched from the memory. Accordingly, it returns with the number of cycles spent in fetching this data. The result is then accumulated in a global variable.

The annotated code is compiled, and run on the host machine. Accurate timing information is made available from the instrumentation. To calculate the power consumed, Power State Model approach is used. The processor is in a switching (running) state when instructions are being executed in the pipeline, and is in idle state while waiting for data to be fetched from the memory. The power consumed by the processor in each stage in unit time is known. By using this information, total power consumed in running the benchmark application on the target processor can be estimated. Similar approach is used for estimating the power consumed in fetching data from the memory.

3.2 Mapping between Source and Binary Code

It is clear that for correct instrumentation accurate mapping between source code and binary code is very important, but this mapping is generally destroyed by the phases of compiler optimizations. Compilers have multiple

4 Implementation

5 Results

6 Conclusion

List of Figures

List of Tables