CSC 714 Real Time Computer Systems Project

Pipeline and Path analysis for Power-aware embedded architecture

FINAL REPORT

-Kiran Seth

Table of Contents

1. Introduction

In Real-Time systems, it is critical to have guaranteed temporal and logical performance. Correct operation requires task deadlines to be met while maintaining correctness of the operation. The Worst Case Execution Time (WCET) for tasks is required for schedulability analysis and also for creating the schedules. A naï ve WCET means that we are losing the opportunity of using slack that is available in the system and also losing the ability to increase the number of tasks admitted into the system.

Another issue in Real Time systems is the conservation of power. When designing the system, the frequency of the processor has to be decided. Higher the frequency, higher is the power consumption. Because of naï ve WCET bounds, we do not know the exact frequency required by the system and significant power is wasted in current Real Time systems due to overclocking. The processing speed or clock frequency of the system is higher than required because a deadline cannot be missed.

Thus both these problems can be tackled with tools that provide us with accurate WCET bounds. In this project I have worked towards modifying existing tools to produce tight WCET bounds for a simple pipeline that executes the Pseudo ISA (PISA) of the Simplescalar toolset. The obvious advantages of such a tool are explained above.

2. Problem Overview

As the deadlines of Real-Time system tighten, performance increasing techniques like caches, pipelining, branch prediction and out of order execution are added to embedded processors. These complex pipelines increase the performance of embedded processors, but it is difficult to guarantee the performance of a complex pipeline. It is very difficult to accurately measure WCET of tasks on a complex pipeline. But for simple pipelines, it is possible to calculate the WCET accurately. Due to the WCET requirements, simple pipelines are preferred in hard Real Time Systems. But simple pipelines cannot show the same performance as a complex pipeline.

A simple solution is to use a dual frequency approach. The hybrid embedded processor will usually work at a low frequency with a complex pipeline. It is expected that the actual execution time required is smaller than the WCET. The tasks are now given intermediate deadlines. If the task misses any intermediate deadline, the frequency is increased. At this time, we can also switch the pipeline from a complex to simple pipeline. This means that since we can guarantee the performance of a simple pipeline, we are assured that no deadline will be missed. The low and high frequencies are chosen depending on the WCET of the task and the actual execution time (which can be found by using simulations of the program). One of the main requirements in this approach is accurate WCET analysis of a program.

The goal of this project is to modify existing tools to provide accurate WCET predictions for the SimpleScalar ISA and then to add parametric timing analysis paradigms to the tools.

3. Previous work

Currently tools are available to perform static instruction cache simulation (work by Dr. Mueller), static data cache simulation (work by Dr. White) and the timing analysis of programs using path analysis and caching categorizations (work by Dr. Healy). The tools take inputs files generated by a research compiler (vpcc/vpo). The timing analyzer also takes input files from the static instruction cache simulator and the static data cache simulator. But we are going to use a modified version of the SimpleScalar simulator (provided by Dr. Rotenberg) to calculate the actual execution times of programs using simulation. This means that all the WCET analysis must also be done for the SimpleScalar ISA (PISA). The static instruction cache simulator is ISA independent. But the timing analyzer currently works with the microSparc ISA. Also, the compiler that generates PISA binaries does not generate the input files for the various tools.

It important to perform accurate WCET of tasks for simple pipelines. The program is first analyzed using a static instruction cache simulator and a static data cache simulator to calculate the caching potential of all the instructions. The instructions are categorized into always hit, always miss, first hit and first miss. To get an accurate WCET, we need to perform path analysis for the given program and look at the interaction between paths. The path analysis along with the caching categorizations are used to predict the WCET of the program. Overlapping of two operations is also taken into account (like a high latency floating point operation and a cache miss may overlap) thus making the WCET more accurate.

4. Current work

The objective of the project is to use the existing framework for timing analysis and use it with a different ISA and pipeline. The old timing analyzer works with the microsparc architecture and it is being ported to the ISA used by the Simplescalar toolset. The old timing analyzer also has a simple pipeline with 7 stages which will be changed to a 6 stage pipeline.

The modifications to the timing analyzer are-

- A new pipeline, compatible with the one used in the modified Simplescalar timing simulator. The new pipeline is a simple pipeline with 6 stages.

- Branch prediction using a static prediction technique (currently Ball Larus heuristic). By using a static branch prediction mechanism, the WCET bounds can be improved because we don't have to add the branch mispredict penalty all the time. Any form of static branch prediction can be used (prediction using profiling, other heuristics, etc.).

- A new query interface. The new query interface makes it easier to get the WCET for parts of the program, instead of only giving the WCET for the whole program.

The first step was to reverse engineer the PISA assembly to produce inputs for all the tools (like the vpcc/vpo compiler). This means, looking at the assembly to form basic blocks and the control flow graph (CFG). The CFG is then used to create inputs for all the tools. After creating a software patch for reverse engineering the assembly, Timing Analyzer's ISA is changed to PISA. The internal pipeline it uses for simulation is also changed so that it looks and works just like the pipeline in the Simplescalar simulator. Thus we will be able to analyze programs compiled using the PISA gcc and determine an accurate and tight WCET for the programs.

5. Experiments

The following table shows the WCET calculated by the timing analyzer and the actual execution time as computed by the Simplescalar tool.. The results are for a 1GHz processor and are converted into time instead of cycles.

Benchmarks	WCET (us)	Actual execution time (us)	WCET/actual execution time
adpcm	3286	2428	1.35
cnt	72	71	1.01
fft	426	368	1.16
lms	173	168	1.03
srt	3508	2050	1.00
mm	2056	1755	2.00

It can be observed that in most cases the WCET timing is very close to the actual execution time. The only discrepancies are seen for benchmarks with conditional statements, which may be inside loops. If one "arm" of the conditional statement is longer than the other, the timing analyzer will assume that the longest path will always be taken while in the simulator may usually take the shorter path.

6. Future work

The next step would be to modify the timing analyzer so that it is parametric in terms of frequency. Currently the timing analyzer gives us the WCET in terms of number of cycles. This means that the timing analyzer must be run again and again for getting the WCET for different frequencies. By parameterizing the output of the timing analyzer in terms of frequency, the timing analyzer can be run only once and the output WCET can be used for any frequency by simply plugging in the frequency. Parameerizing the timing analyzer will also make it more power aware.

Another tool that currently cannot be used with the timing analyzer is the data cache simulator. A part of the future work could be integrating the data cache simulator analysis components into the timing analyzer. The timing analyzer would then be able to take data access categorizations into account in giving a tighter and more accurate WCET.

7. References

[1] C. A. Healy, R. D. Arnold, F. Mueller, D. Whalley, and M. G. Harmon. Bounding pipeline and instruction cache performance. *IEEE Transactions on Computer*s, 48(1):53–70, January 1999.

[2] E. Vivancos, C. Healy, F. Mueller and D. Whalley. Parametric Timing Analysis. ACM SIGPLAN Workshop on Languages, Compilers and Tools for Embedded Systems, ACM SIGPLAN Notices, Aug 2001, pages 88-93

[3] F. Mueller. Timing Analysis for Instruction Caches. Real-Time Systems Journal, Vol. 18, No.2/3, May 2000, pages 209-239

[4] R. White, F. Mueller, C. Healy, D. Whalley and M. Harmon. Timing Analysis for Data Caches and Set-Associative Caches. Real-Time Technology and Applications Symposium, Jun 1997, pages 192-202

[5] D. Burger, T. Austin and S. Bennett. Evaluating Future Microprocessors: The Simplescalar Toolset. Technical report CS-TR-96-1308, Computer Sciences Department, University of Wisconsin-Madison, July 1996.

[6] E. Rotenberg. Using variable-Mhz microprocessors to efficiently handle uncertainty in real-time systems. *34th International Symposium on Microarchitecture*, December 2001.

[7] Thomas Lundqvist. "A WCET Analysis Method for Pipelined Microprocessors with Cache Memories," PhD thesis, Dept. of Computer Engineering, Chalmers University of Technology, Sweden, June 2002.

[8] Jakob Engblom. Effects of Branch Predictors on Execution Time. Dept. of Information Technology Technical Report 2002-013, April 2002.