FAST INSTRUCTION CACHE ANALYZER SOFTWARE TOOL

Nikolaos Kroupis[1], Alexandros Bartzas[1], Stylianos Mamagkakis[2], Dimitrios Soudris[1]

nkroup@ee.duth.gr

[1] *Department of Electrical and Computer Engineering – Democritus University of Thrace – Greece*
[2] *IMEC vzw, Leuven, Belgium*
http://vlsi.ee.duth.gr/

Abstract

The Fast Instruction Cache Analyzer Tool explores the instruction cache memory of DSP applications within reasonable time and high accuracy, for variable cache parameters, without simulation.

1. Introduction

The main goal of the tool is to provide fast and accurate estimates of the number of (-micro) instructions and the instruction cache miss rate on a programmable embedded platform, during the early design phases. An automatic software tool named Fast Instruction Cache Analyzer (FICA) is developed using the programming language PHP using a MySQL database to store its data. We have selected PHP and MySQL because we intend to provide a web-based tool, which will be available to the researchers. FICA uses specific information which is extracted from both the high-level code description (C code) of the DSP application and its corresponding assembly code, without carrying out the time-consuming procedure of simulation. FICA requires only a single execution of the application in a general-purpose processor and uses only the assembly code of the targeted embedded processor.

2. FICA Software Tool

The flow graph of the tool is presented in Fig. 1. The tool has as input only the application's code in C language and it corresponding assembly code of the specific programmable core. FICA is based on the correlation between the high-level description code of the application and its associated assembly code. The crucial point of the tool is that the number of conditional branches both the C code and its assembly code is equal. Fig. 2 presents the processing steps of FICA tool using a simple application (input) C code and it's corresponding assembly. First stage of the tool pinpoint the code branches and inserts counters in C code. Executing the new C code in general purpose system, FICA determines the number of passes from every branch. This a very fast procedure, because the execution is been in general purpose system and is a platform independent stage. The extracted execution data are stored into a database.
In the second stage, based on branch and jump instructions of the assembly code, the tool creates the Control Flow Graph (CFG) of the assembly code. The crucial point of the tool is to assign the number of each branch execution to the specific node of the CFG. The values correspond to the assembly code, and thus it is finds how many times each assembly branch instruction is executed. Using a exploration technique, the tool calculate the number of executions of every node of CFG. Adding the number of execution of all application's assembly instructions the total number of executed instructions is calculated in the second stage. Such, the number of instructions is estimated without simulation, executing only once the application in a general purpose system.

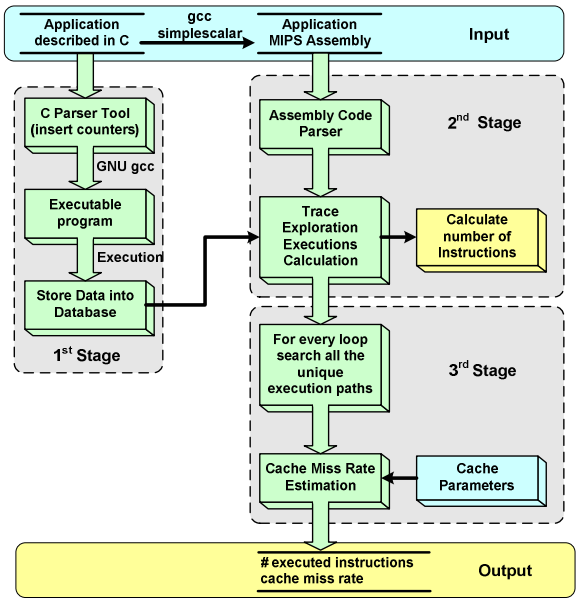

Figure 1: The flow of FICA tool

The third stage of the tool is platform-dependent and contains two steps: the creation of all the unique execution paths of each loop, and the computation of number of instructions and iterations associated with a unique path. Exploring all the paths of the CFG of the algorithm, we determine the loops and the size (in numbers of instructions), as well as the number of executions of each loop. Comparing the length of the path with the size of the cache, it calculates the number of misses of each every loop iteration. The FICA tool derives the instruction cache miss rate for variable cache sizes and block sizes with a single execution. Moreover, the tool presents all combinational results of miss rates using variable parameters of L1 and L2 instruction caches.

3. Experimental Results

We adopt MIPS IV processor 64-bit running on simulation tool. In order to evaluate the developed tool, we chose a set of benchmarks from various digital signal processing

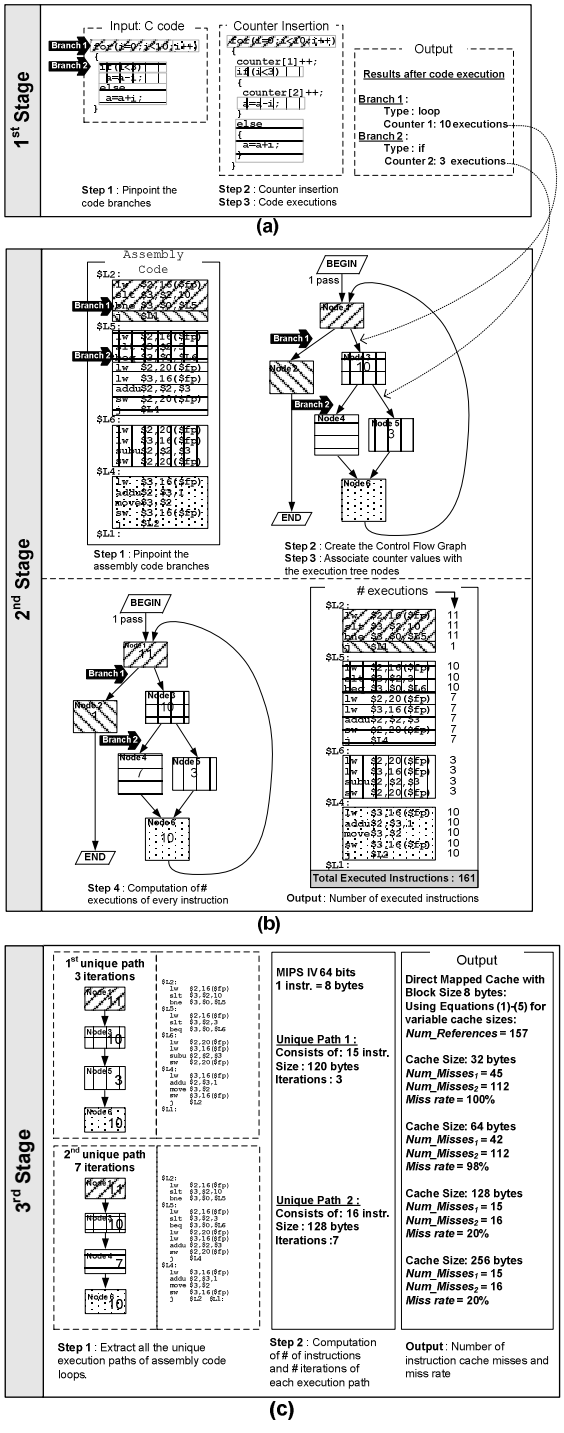

Figure 2: The details steps of the FICA tool, using a simple input code

applications, such as MPEG-4, JPEG, filtering and H.263. The comparison between the results of FICA and the simulation results is given by the average estimation error which is about 3% for the number of executed instructions. Assuming variable cache architectures (with L1 and L2 caches) and configurations, with variable cache and block sizes, the estimation results of miss rate show that the average estimation error is about 5% comparing with the simulation results. Such, FICA exhibits high accuracy in estimation of number of executed instructions and number of cache misses and miss ratio. An additional crucial feature of a high-level estimation procedure is the fact that the proposed approach is faster thousands of times than the corresponding simulation procedure. This feature enables the designers to explore the cache configuration within a reasonable time (multi-level caches). Table I presents the comparison results between estimated and simulator measurements for variable cache sizes.

Table I. Instruction cache miss rate using the simulator Simplescalar and the estimation tool FICA

Miss Rate Block size 8 bytes		Cache Size 128	256	512	1K	2K	4K	Aver. error
FS	**Simplesc.**	100.0	99.8	99.2	76.8	0.1	0.0	
	FICA	100.0	99.9	99.6	71.9	0.1	0.0	0.9 %
HS	**Simplesc.**	97.3	92.5	66.4	2.8	1.6	1.5	
	FICA	96.0	87.5	60.8	3.4	3.2	0.5	2.5 %
3SLOG	**Simplesc.**	99.7	93.1	15.9	1.9	1.8	0.0	
	FICA	99.4	96.9	7.8	0.9	0.5	0.0	2.4 %
PHODS	**Simplesc.**	99.9	99.6	96.7	31.7	0.8	0.2	
	FICA	100.0	98.8	96.1	22.7	1.0	1.0	1.8 %
SS	**Simplesc.**	99.9	98.9	79.9	0.5	0.1	0.0	
	FICA	99.2	98.4	75.0	0.1	0.0	0.0	1.1 %
Wavelet	**Simplesc.**	90.0	47.9	1.8	1.0	0.1	0.0	
	FICA	92.7	43.3	0.4	0.2	0.0	0.0	1.6 %
Cavity Detector	**Simplesc.**	100.0	94.3	61.4	16.9	0.3	0.1	
	FICA	100.0	94.6	45.7	5.1	0.2	0.1	4.7 %
CQ	**Simplesc.**	99.4	89.1	46.5	9.6	0.4	0.0	
	FICA	98.8	84.2	31.4	2.0	0.0	0.0	4.8 %
FFT	**Simplesc.**	98.7	95.7	87.7	7.1	0.5	0.2	
	FICA	100.0	96.0	75.3	9.4	6.0	6.0	4.6 %

4. Conclusion

Concluding, the total estimation time cost using FICA tool is much smaller than that obtained by the trace-driven techniques and simulations. Compare FICA and simulation based approaches is faster in a factor more than 100 times. FICA is suitable for performing high-level instruction cache exploration with a very high accuracy reducing the time cost of the simulation. FICA estimates the cache miss rate, without the simulation of the application and reduces the estimation time by orders of magnitude compared to the simulation process.

5. References

[1] N. Kroupis, S. Mamagkakis, and D. Soudris, “An Estimation Methodology for Designing Instruction Cache Memory of Embedded Systems,” in ESTIMedia 2006, Fourth IEEE Workshop on Embedded Systems for Real Time Multimedia, October 26-27, 2006, Seoul, Korea

Acknowledgement

The project is co-funded by the European Social Fund & National Resources - EPEAEK II - PYTHAGORAS II.