Using QEMU in timing estimation for mobile software development

Antti P. Miettinen¹, Vesa Hirvisalo², and Jussi Knuutila²

¹ Nokia Research Center, Finland, antti.p.miettinen@nokia.com
² Aalto University, Finland, firstname.lastname@tkk.fi

Abstract. We present our research on using QEMU as a platform for software development that targets mobile hand-held devices. For such devices, understanding the timing of software execution is essential for energy consumption estimation. Traditional cycle-accurate simulators are orders of magnitude slower than real hardware, and thus unsuitable for software developers. Our research shows that embedding an abstract hardware simulator within QEMU achieves simulation times and accuracy required for software development.

1 Introduction

Software development targeting mobile devices is by its nature cross-development. The use of functional simulators in such development environments is an industry standard practice. High simulation speed is crucial for the simulators used in these setups as the productivity of the software development is greatly affected by the speed of iteration associated with contemporary software engineering practices.

However, understanding power and performance characteristics of the software is very difficult when the simulation is only functional. The simulation time on a development host can be a severely misleading indicator for the performance of the software on a real target device. For mobile devices, especially the energy consumption of the software is a critical metric of interest. Considering modern hardware and software, energy consumption can be estimated if the timing of the execution can be simulated [1]. Our research has addressed simulating the timing behavior of software by using QEMU as our research tool.

2 Our contribution

Our research contribution is fitting a practical timing simulator of modern multi-core hardware within a functional emulator without a significant performance loss [2]. Our modified version of QEMU, called pQEMU [3],
Table 1: Test platforms and simulation overhead

<table>
<thead>
<tr>
<th>Platform</th>
<th>CPU</th>
<th>Speed</th>
<th>Cache (L1, L2)</th>
<th>Instrumentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>PB11MPCore</td>
<td>ARM11</td>
<td>210MHz</td>
<td>32k+32k, 1M</td>
<td>none</td>
</tr>
<tr>
<td>Naviengine NE1</td>
<td>ARM11</td>
<td>400MHz</td>
<td>32k+32k, none</td>
<td>classify</td>
</tr>
<tr>
<td>Beagleboard C3</td>
<td>Cortex-A8</td>
<td>500MHz</td>
<td>16k+16k, 256k</td>
<td>0.3</td>
</tr>
<tr>
<td>Beagleboard C4</td>
<td>Cortex-A8</td>
<td>720MHz</td>
<td>16k+16k, 256k</td>
<td>1.0</td>
</tr>
<tr>
<td>KZM</td>
<td>Cortex-A9</td>
<td>500MHz</td>
<td>32k+32k, 512k</td>
<td>0.9</td>
</tr>
<tr>
<td>Tegra</td>
<td>Cortex-A9</td>
<td>1GHz</td>
<td>32k+32k, 1M</td>
<td>2.4</td>
</tr>
</tbody>
</table>

simulates the timing of instruction executions and memory latencies. Instruction execution timings are simulated using instruction classification and weight coefficients, while memory latency is simulated using a set-associative cache and TLB simulator.

In order to keep the overhead of the simulation as low as possible, we perform multiple optimizations. These include instrumentation at basic block granularity using direct TCG load-add-store triplets and SIMD vectorized cache simulation. We have also experimented with using multicore parallelization for running the cache simulation in parallel, but we found it to offer relatively modest speedups at significant implementation difficulty.

While highly accurate hardware modelling can cause a dramatic increase in simulation overhead, reasonable accuracy can still be achieved with a lightweight simulation. Our method is based on calibrating the

Fig. 1: Estimation error and simulation overhead with optimizations

(a) Estimation error

(b) Simulation optimizations
Fig. 2: Comparing performance scaling of x264 with different video frame sizes in real hardware and inside simulation.

simulator with timing measurements from real hardware. Table 1 shows our test platforms and illustrates how the simulation overhead increases in our non-optimized implementation when more details are added to the simulation. Conversely, Figure 1a illustrates how the timing estimation error is affected by the simulation detail. The effect of our optimizations on the overhead is shown in Figure 1b.

3 Related work

Traditional cycle accurate simulators are typically several orders of magnitude slower than real hardware, whereas purely functional simulators can almost reach the speed of the simulated hardware. This has led to the development of cycle-approximate instruction set simulators [4, 5].

Extending functional simulators with timed models is often motivated by the needs of hardware development [6, 7]. As our concern is software development, simulation performance is a prime concern. Instead of accurate hardware modelling, we aim towards hardware abstraction and rely on calibration for getting the accuracy to an acceptable level.

4 Future work

Current QEMU system mode emulation does not utilize host cores for simulating the target cores in parallel. As the number of processor cores is rising in cell phones, laptops, etc., we see this direction of research
and development vitally important for the future. Such a research and
development effort is already taking place in the COREMU project \[8\].

An interesting topic for future research is whether execution driven
simulation is able to provide meaningful performance estimates for mul-
tithreaded workloads while maintaining high simulation speed. We have
done preliminary studies on the topic. Figure 2 show results of our exper-
iment, where the performance scaling of x264 inside COREMU is com-
pared to the scaling in real hardware. To better understand the appli-
cability of QEMU, more experimentation is needed with true multicore
emulation.

References

1. Miettinen, A., Hirvisalo, V.: Energy-efficient parallel software for mobile hand-held
2. Miettinen, A., Hirvisalo, V., Knuuttila, J.: Execution-driven simulation of non-
Conference (ESM). (2010)
4. Cmelik, B., Keppel, D.: Shade: a fast instruction-set simulator for execution profil-
SW/SystemC SoC Emulation Framework. In: Proc. IEEE Int. Symposium on In-
dustrial Electronics (ISIE). (2007)
7. Gligor, M., Fournel, N., Pétrot, F.: Using binary translation in event driven simula-
on Hardware/software codesign and system synthesis, New York, USA (2009)
Institute, Fudan University.