Software Co-Verification Based on Program Traces from Different Processors

Rodolfo J. Azevedo (PQ), Luís F. Antonioli (IC).

Abstract
The ever increasing complexity of electronic systems and tight time-to-market has made designers of MPSoC (Multiprocessor System-on-chip) adopt higher abstraction levels and the so called Electronic System Level (ESL) design methodology. In this scenario, virtual platforms play an important role. Processors are key components of such virtual platforms, where Architecture Description Languages are applied to automatically generate simulators and other software tools. Considering the importance of virtual platforms verification, co-verification mechanisms based on matching execution traces are going to be studied in this project, focusing on comparing the execution trace of the same program in two distinct architectures.

Key words: Co-Verification, Execution traces, Distinct architectures.

Introduction

Architecture description languages (ADLs) has been widely used to build processors models. They can generate software tool chain, like assemblers, linkers, compilers, debuggers, simulators, etc, which makes them valuable in design automation methodology. Therefore, it is very common to ADL processor models to be included into high-level virtual platform models. This project focuses on the software co-verification path. Our objective is build a high-level tracing generation feature for the ArchC[1,2] generated simulators and a verification system that can match the execution trace from distinct processors providing a good feedback about the correct execution of a program.

ArchC is an Architecture description language that uses a simple processor definition as input and generates several tools as output like assemblers, linkers, debuggers, and ISS. The generated ISS is based on the SystemC[3] simulator, allowing easy integration with external components through a TLM[4] interface.

Results and Discussion

Since our verification system works in a high abstraction level, it uses smaller trace files as input when compared to the sequence of PC addresses that the simulator executes in the execution of a program. In the table 1 we compare the size of these two types of trace in four different programs from the Mibench Benchmark using an ArchC generated simulator for the MIPS processor.

Table 1. The size of the traces generated using the MIPS simulator and four programs from the Mibench benchmark.

<table>
<thead>
<tr>
<th>Program</th>
<th>Low-level trace</th>
<th>High-level trace</th>
</tr>
</thead>
<tbody>
<tr>
<td>Qsort</td>
<td>196.3 MB</td>
<td>3.1 MB</td>
</tr>
<tr>
<td>Rijndael</td>
<td>229.8 MB</td>
<td>6.5MB</td>
</tr>
<tr>
<td>SHA</td>
<td>51.4 MB</td>
<td>3.2MB</td>
</tr>
<tr>
<td>Susan</td>
<td>15.4MB</td>
<td>8.1MB</td>
</tr>
</tbody>
</table>

Conclusions

This project provided an approach to co-verify a program execution by matching the execution trace from the current architecture against another one that has already a correct execution environment. We developed a new feature into the ArchC project to generate simulators capable of generate a high-level execution trace of a program based on its source code debug information and a verification system that provides a good feedback about the correct execution when comparing two distinct processors.

Acknowledgement

I would like to thank my advisor Prof. Rodolfo Azevedo for the continuous support, Maxiwell Garcia for the patience as well as my family and friends.

This project was supported by FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo)