Cycle Approximate Simulation of RISC-V Processors

Lee Moore, Duncan Graham, Simon Davidmann
Imperas Software Ltd.
Oxford, United Kingdom
simond@imperas.com

Felipe Rosa
Universidad Federal Rio Grande Sul
Brazil

***Abstract*— Historically, architectural estimation, analysis and optimization for SoCs and embedded systems has been done using either manual spreadsheets, hardware emulators, FPGA prototypes or cycle approximate and cycle accurate simulators. This precision comes at the cost of performance and modeling flexibility.**

Instruction accurate simulation models in virtual platforms, have the speed necessary to cover the range of system scenarios, can be available much earlier in the project, and are typically an order of magnitude less expensive than cycle approximate or cycle accurate simulators. Previously, because of a lack of timing information, virtual platforms could not be used for timing estimation. We report here on a technique for dynamically annotating timing information to the instruction accurate software simulation results. This has achieved an accuracy of better than +/-10%, which is appropriate for early design architectural exploration and system analysis. This Instruction Accurate + Estimation (IA+E) approach is constructed by using Open Virtual Platforms (OVP) processor models plus a library that can introspect the running system and calculate an estimate for the cycles taken to execute the current instruction. Not only can these add-on libraries dynamically inspect the running system estimate timing effects, they can annotate calculated instruction cycle timing back into the simulation and affect timing of the simulation.

Keywords—RISC-V, virtual platform, instruction accurate, processor models, timing estimation

I. INTRODUCTION

Performance and power consumption are two key attributes of any SoC and embedded system. Systems often have hard timing requirements that must be met, for example in safety critical systems where reaction time is of paramount importance. Other systems, particularly battery powered systems, have power consumption limitations.

Because of the importance of these characteristics, many techniques have been developed for estimation of performance and power consumption. Recently, with the explosion of system scenarios that must be considered, this job has become much more difficult.

Instruction accurate simulation has previously not been considered as a potential technique for timing and power estimation, because it is instruction accurate and does not model processor microarchitecture details: there is no information about timing or power consumption of instructions and actions in instruction accurate models and simulators. Recently some universities, using the Open Virtual Platforms (OVP) models and OVPsim simulator [1], have experimented with adding this information into the instruction accurate simulation environment as libraries, with no changes to the models or simulation engines [2]. These efforts have shown great promise, with timing estimation results within +/- 10% of the actual timing results for the hardware for limited cases.

We report here on the further development of this technique, and the extension of this technique for RISC-V ISA based processors. This is critical for the RISC-V ecosystem, since for RISC-V semiconductor vendors to win embedded system sockets, their customers are going to want to know about the timing and power consumption of those SoCs when running different application software.

II. CURRENT STATE OF THE ART

Historically, SoC architectural estimation, analysis and optimization has been done using either manual spreadsheets, hardware emulators, FPGA prototypes, cycle approximate simulators or cycle accurate simulator and performance simulators such as Gem5 [3]. These all have significant drawbacks: insufficient accuracy, high cost, RTL availability (meaning that the technique is only available later in the project when the RTL design is complete), low performance, limited ability to support a wide range of system scenarios or are very complex to use and gain good results. Table 1 provides a summary of the strengths and weaknesses of each technique.

TABLE I. STRENGTHS AND WEAKNESSES OF CURRENTLY USED TECHNIQUES FOR TIMING AND POWER ESTIMATION

Technique	Strength	Weaknesses
Manual spreadsheets	Ease of use	Lack of accuracy; inability to support estimations with real software
Hardware emulators	Cycle accurate	High cost (millions USD); needs RTL; < 5 mips performance
FPGA prototypes	Cycle accurate	High cost (hundreds of thousands USD); needs RTL
Cycle approximate simulation	Good performance	Lack of accuracy; lack of availability of models
Cycle accurate simulation	Cycle accurate	High cost (hundreds of thousands of USD); lack of availability of models
Gem5	Microarchitectural detail	A lot of work to develop a model of specific microarchitecture and to get realistic traces of SoC.

III. INSTRUCTION ACCURATE SIMULATION

Instruction set simulators (ISSs) have long been used by software engineers as a vehicle for software development. Over the last 20 years, this technique has been extended to support not only modeling of the processor core, but also modeling of the peripherals and other components on the SoC. The advantages of these simulators are their performance, typically hundreds of millions of instructions per second (MIPS), and the relative ease of building the necessary models. However, the simulator engines and models are instruction accurate, and are not built to support timing and power estimation.

The performance of these simulators comes from the use of Just-In-Time (JIT) binary translation engines, which translate the instructions of the target processor (e.g. Arm) to instructions on the host x86 PC. This enables users to run the same executables on the instruction accurate simulator as on the real hardware, such that the software does not know that it is not running on hardware. Peak performance with these simulators can reach billions of instructions per second. A more typically use case, such as booting SMP Linux on a multicore Arm processor, takes less than 10 seconds on a desktop x86 machine.

There are also significant libraries of models available, and it is easier to build instruction accurate models than models with timing or power consumption information, or real implementation details. One such library and modeling technology is available from OVP. The OVP processor model library includes models of over 200 separate processors (e.g. Arm, MIPS, Power, Renesas, RISC-V), plus a similar number of peripheral models. Most of these models are available as open source. The C APIs for building these models are also freely available as an open standard from OVP.

IV. INSTRUCTION ACCURATE SIMULATION PLUS ESTIMATION

Instruction accurate simulation holds the promise of faster simulation performance to support examination of more system scenarios, plus lower cost and earlier availability. With the Imperas APIs and dynamic model introspection it is easy to add in timing and power estimation capabilities into the instruction accurate simulation environment.

The idea of adding these capabilities as libraries is the combination of annotation techniques and binary interception libraries used with JIT simulation engines. Annotation techniques can be imagined as a full instruction trace which is then annotated with the timing or power information. However, just using annotation requires significant host PC memory, and can slow the simulation.

Binary interception libraries are used with the Imperas JIT simulators to enable the non-intrusive addition of tools, such as code coverage and profiling, to the simulation environment. Combining these techniques maintains the high simulator performance with minimal memory costs. This combined technique is being called Instruction Accurate + Estimation (IA+E).

In the Imperas simulation products, which require the use of OVP models, it is possible to create a standalone library module with entry points that are called when instructions are executed. This library can introspect the running system and calculate an estimate for the cycles taken to execute the current instruction, and can take into account overhead of different memory and peripheral component latencies. Not only can these add-on libraries dynamically inspect the running system and estimate timing affects, they can annotate calculated instruction cycle timing back into the simulation and affect (i.e. stretch) timing of the simulation. An overview of the simulation architecture is shown in Figure 1.

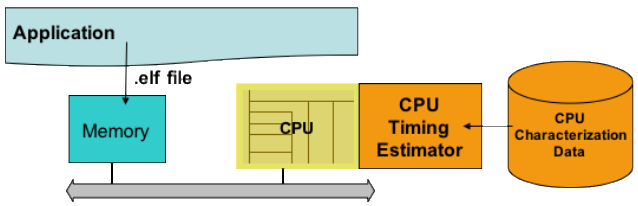

Fig. 1. Overview of the Imperas IA+E simulation environment.

For processors, the instruction estimation algorithm includes:

- a mixture of table look ups for simple instructions
- dynamic calculations for data dependent instructions
- adjustments due to code branches taken
- taking into account effects of memory and register accesses

A view of the timing estimation mechanism is shown in Figure 2.

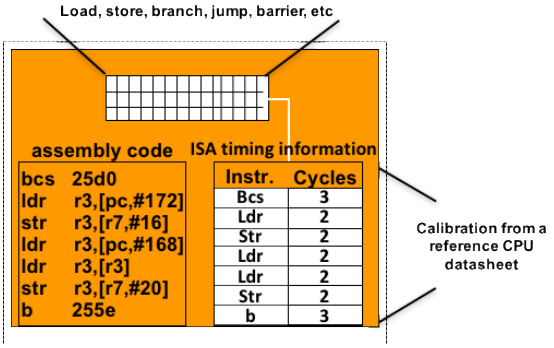

Fig. 2. Simplified view of the timing estimation mechanism.

For memory subsystems and peripheral components table, lookup and dynamic estimation can be made and timing back annotated into the simulation to simulate the delay effects of slow memories and other components.

With this Instruction Accurate + Estimation (IA+E) approach, there is a separation of processor model functionality and timing estimation. This means while building a functional model there is no need to worry about any timing or cycle complexity. It is only when the more detailed timing is needed is it necessary to add the extra timing data to enable the Imperas IA+E timing tools to provide cycle approximate timing simulation for the RISC-V processors.

This extra timing data is added in two steps. First, the cycle information is added to the library. Second, the time per cycle, which is dependent upon the specific semiconductor process and physical implementation details, is added.

The approach of providing the timing data as a separately linked dynamic program enables RISC-V processor designers to create a cycle approximate timing simulation for their specific processor implementation - without sharing any internal information.

IA+E simulation performance slows down from normal simulation performance, with typical overhead of about 50% of normal performance. Still, this puts IA+E simulation performance at 100-500 MIPS.

IA+E does have some limitations. This technique has currently been proven only for simple processors with a single core, no cache, and in-order pipeline.

V. Results

This IA+E technique was first tested with Arm Cortex-M4 based processors. The results were much better than expected, with an average estimation error of +/- 5% as compared to the actual device. The device was an ST Microelectronics STM32F on a standard development board, running the FreeRTOS real time operating system, with 39 different benchmark applications used. Almost all timing estimation errors were within +/- 10% of actual timing values. Figure 3 shows these results.

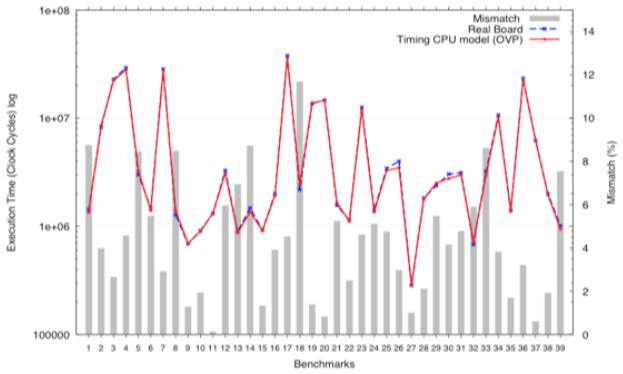

Fig. 3. Timing estimation results for IA+E simulation show average errors of better than +/- 5% over 39 different benchmarks for Arm Cortex-M4.

IA+E was recently extended to support RISC-V processors, by using publicly available information (from the processor vendor's data books) to build the cycle data libraries.

In the data below, showing processor implementations from Andes Technology, Microsemi and SiFive, only the cycle data is presented, since comparing timing for the various implementations would not be an accurate comparison. Also, in keeping with this theme, different benchmark applications were used for each of the different processors. All benchmarks were run with the range of compiler optimization settings, and estimated cycles were reported first assuming 1 cycle per instruction, i.e. using IA, then using the IA+E technique. These results are shown in Figs. 4-6.

VI. Conclusions

The Instruction Accurate + Estimation (IA+E) technique developed here has shown excellent results for timing estimation of in-order processors. It also has the benefits of easy model building, high performance to enable examination of multiple benchmarks and system scenarios, and lower cost than other techniques. In this paper, the IA+E technique has been extended to support RISC-V processors. Further work is needed to apply this technique to power estimation, and to more complex processors.

Acknowledgments

The authors would like to thank Andes Technology, Microsemi, and SiFive for access to their processor datasheets and/or databooks.

References

[1] Open Virtual Platforms (OVP), www.OVPworld.org

[2] Felipe Da Rosa, Luciano Ost, Ricardo Reis, Gilles Sassatelli. Instruction-Driven Timing CPU Model for Efficient Embedded Software Development Using OVP. *ICECS: International Conference on Electronics, Circuits, and Systems*, Dec 2013, Abu Dhabi, United Arab Emirates.

[3] Gem5, www.gem5.org

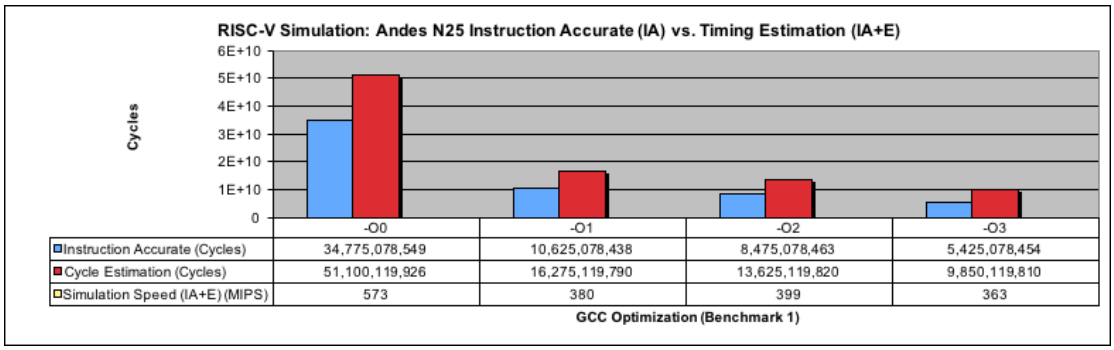

Fig. 4. IA+E cycle estimation results for the Andes N25 processor.

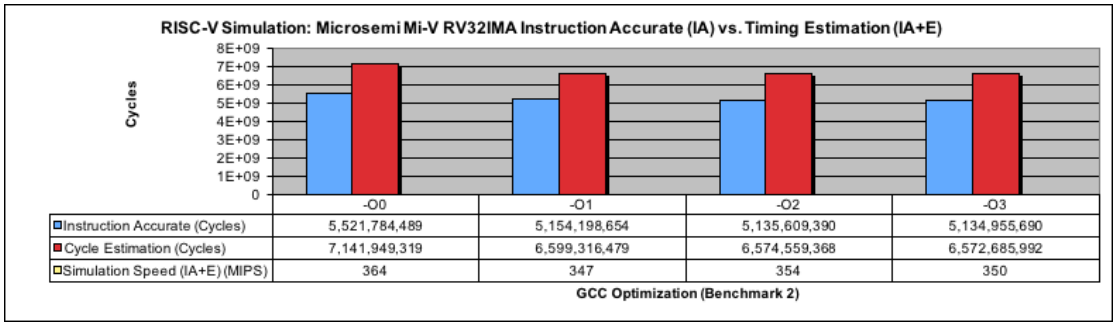

Fig. 5. IA+E cycle estimation results for the Microsemi Mi-V RV32IMA processor.

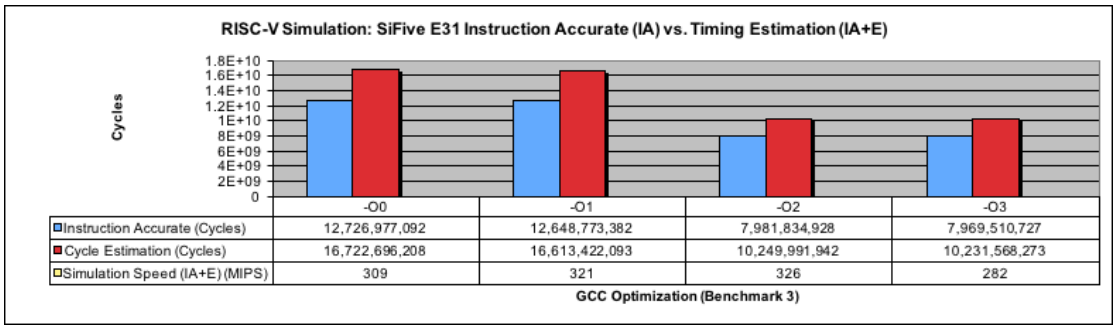

Fig. 6. IA+E cycle estimation results for the SiFive E31 processor.