Cycle Approximate Simulation of RISC-V Processors

Lee Moore, Duncan Graham, Simon Davidmann
Imperas Software Ltd.

Felipe Rosa
Universidad Federal Rio Grande Sul

Embedded World conference
27 February 2018

Agenda

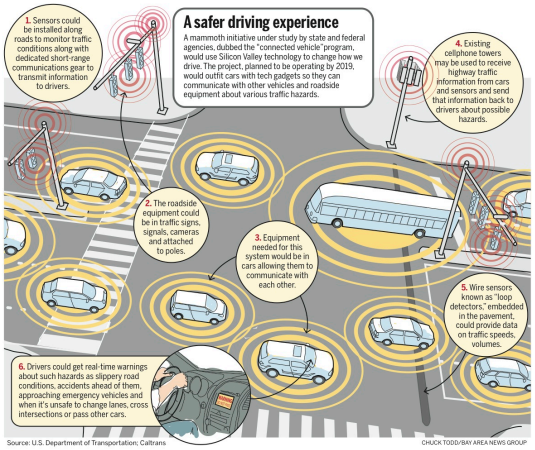
- Why is timing estimation important for embedded systems?
- Current techniques for timing estimation
- Instruction accurate simulation
- Instruction Accurate + Estimation (IA+E)
- Results

Agenda

- **Why is timing estimation important for embedded systems?**
- Current techniques for timing estimation
- Instruction accurate simulation
- Instruction Accurate + Estimation (IA+E)
- Results

Timing Estimation is Important

- Most embedded systems have some sort of real time requirements, e.g. response time limits
- Clearly needed for automotive systems, but also for other transportation systems, medical electronics, industrial controls, IoT, …
- Timing estimation can help with system testing, however this could also help to demonstrate to a prospective customer that the SoC under consideration could meet the system requirements



Challenge

- More and more software in electronic products, with more and more system scenarios, means more and more simulation is needed … which means more and more simulation speed is needed

- So if more timing detail is needed then it needs to be with as much speed as possible
  - Many applications take many millions of instructions

Agenda

- Why is timing estimation important for embedded systems?
- **Current techniques for timing estimation**
- Instruction accurate simulation
- Instruction Accurate + Estimation (IA+E)
- Results

Techniques for Timing and Power Estimation

imperas

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.

RISC-V

Cycle Approximate, Cycle Accurate Simulation Market Survey – several approaches over the years …

- RTL (SystemVerilog) simulation
  - Expensive, slow, late in project, restricted access to IP
- Cycle accurate simulation (RTL based)
  - RTL converted to C, compiled and simulated
  - Expensive, slow, late in project, complex to set up
- Cycle approximate models (C based)
  - Hand coded models…
  - Complex to create models, expensive to build, slowish
- Cycle accurate performance simulation (open source)
  - SimpleScalar, gem5
  - Limited processor architectures, standalone, slow
  - On your own …, support?
- Cycle accurate performance simulation (proprietary)
  - Users who develop their own
  - Requires expert resources, maintenance
  - Usability? Interfaces to standards? Speed? (slow)
  - Traditionally performance simulation has been done with ‘trace based’ solutions as a separate post simulation process that uses large memory and other resources

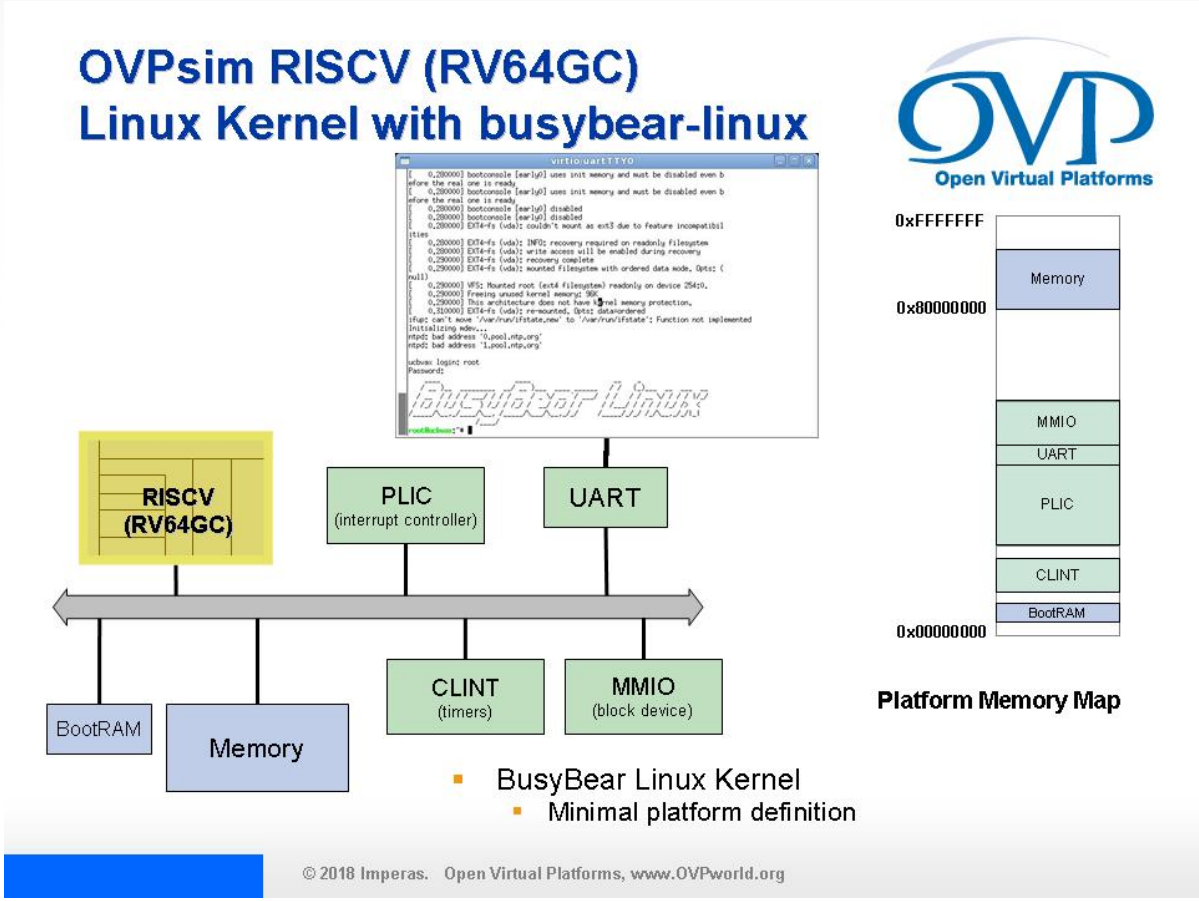
Agenda

- Why is timing estimation important for embedded systems?
- Current techniques for timing estimation
- **Instruction Accurate simulation**
- Instruction Accurate + Estimation (IA+E)
- Results

Virtual Platforms Provide a Simulation Environment Such That the Software Does Not Know That It Is Not Running On Hardware

- The virtual platform is a set of instruction accurate models that reflect the hardware on which the software will execute
  - Could be 1 SoC, multiple SoCs, board, system; no physical limitations
- Run the executables compiled for the target hardware
- Models are typically written in C or SystemC
- Models for individual components – interrupt controller, UART, ethernet, … – are connected just like in the hardware
- Peripheral components can be connected to the real world by using the host workstation resources: keyboard, mouse, screen, ethernet, USB, …
- High performance: 200 – 500 million instructions per second typical, or boots Linux in <5 sec

Example RISC-V IA Virtual Platform



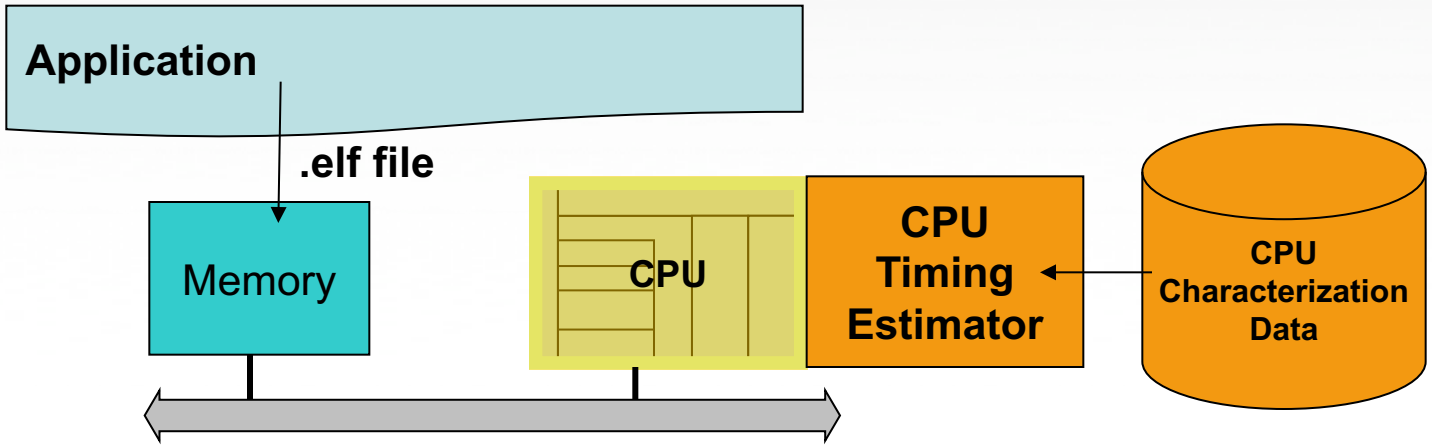
- Boots Linux in under 5 seconds

video

Agenda

- Why is timing estimation important for embedded systems?
- Current techniques for timing estimation
- Instruction accurate simulation
- **Instruction Accurate + Estimation (IA+E)**
- Results

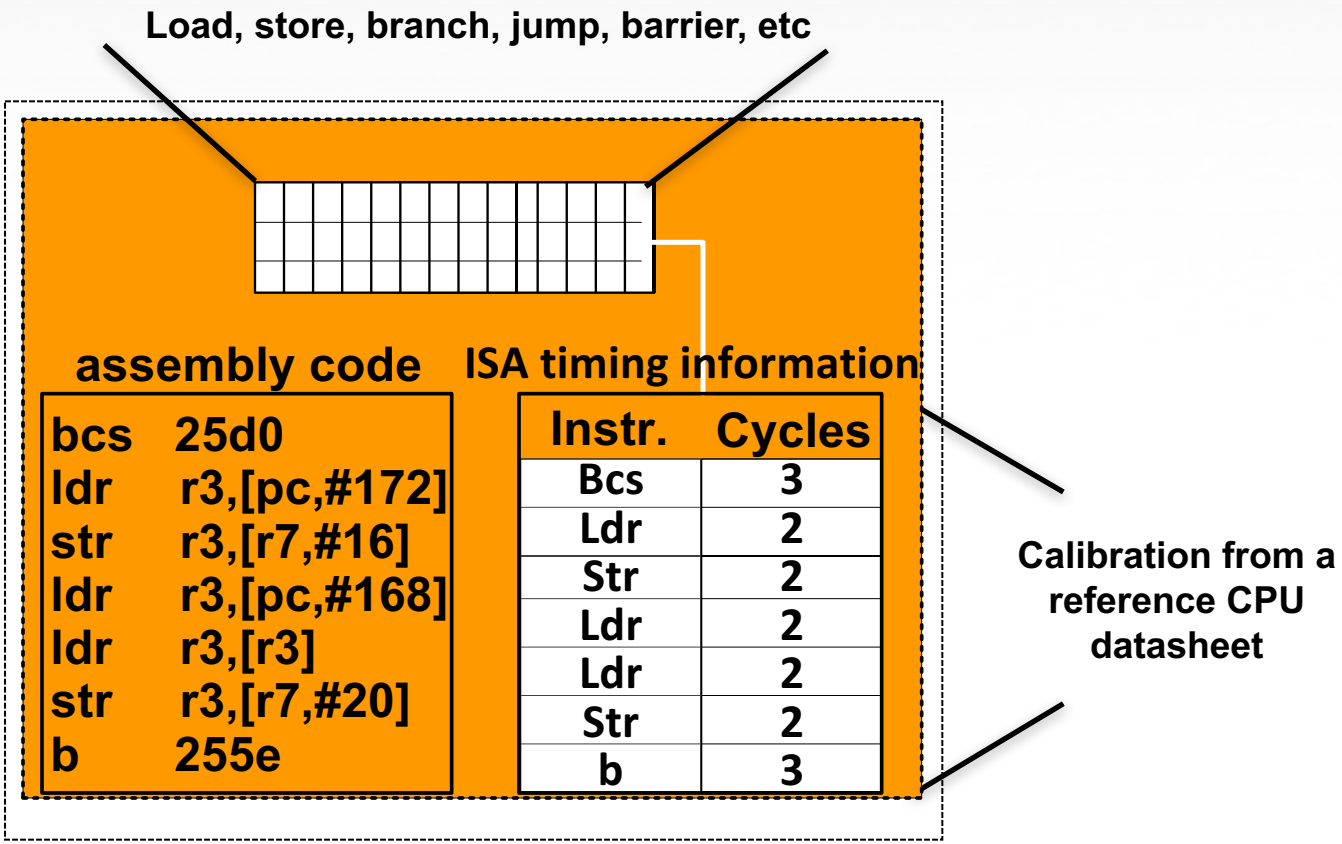
Overview of CPU Timing Estimator



> platform.exe –timing
> platform.exe –timing --stretch

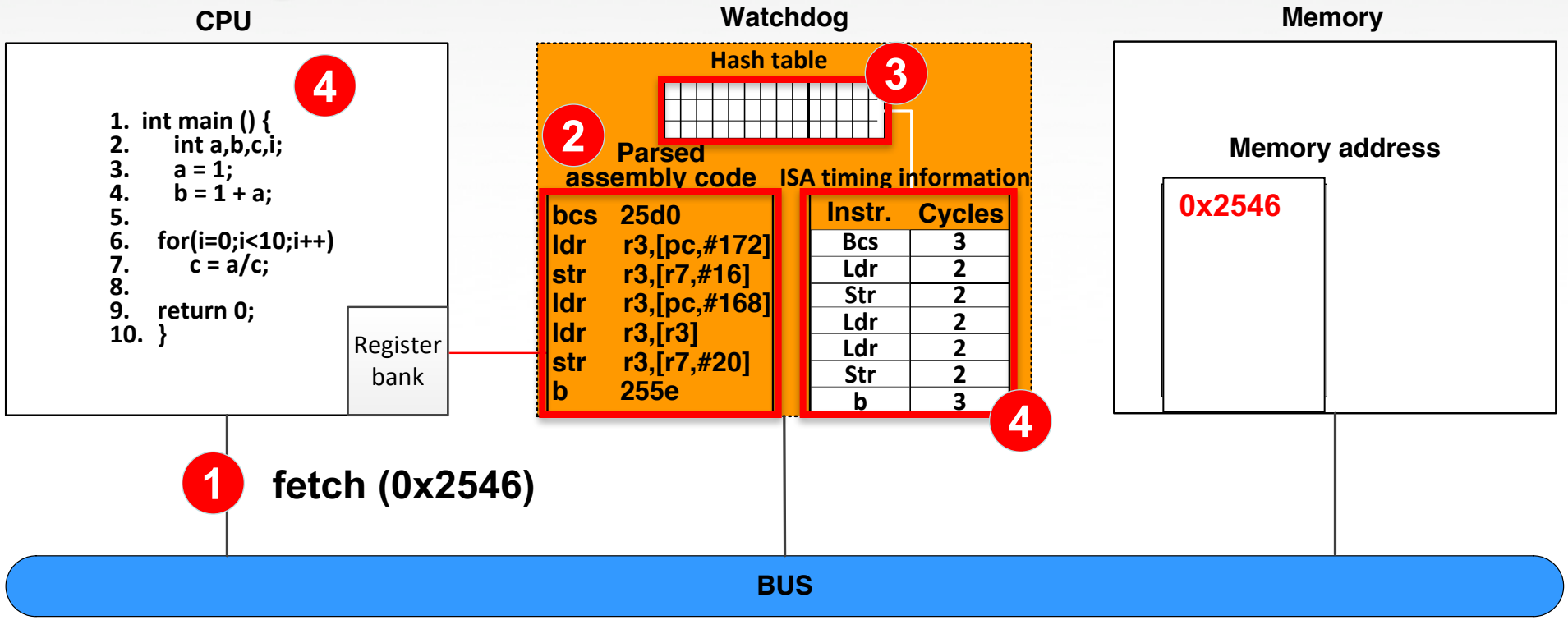
- CPU Characterization Data for each CPU variant
- Timing Estimator loaded onto CPU instance as binary intercept library (SlipStreamer API)
- No edits/changes needed in CPU model binary, or platform, or other models
- Controlled by simulation command line arguments
- Imperas simulation speeds up to 500Mips with timing estimation
  - Note that this is 200-500x faster than callback method with instruction accurate simulation

Mechanism



- Depending on instruction, cycles are added
- Simulator has mode to change elapse of simulation time based on timing calculations (--stretch)

Timing Model



2 the parser module disassembles the binary code and identifies the instruction that must be executed

3 identified instruction is used as a hash table key to ascertain to which class such instruction belongs

4 cycle count is computed and instruction is executed in the CPU

Timing Data

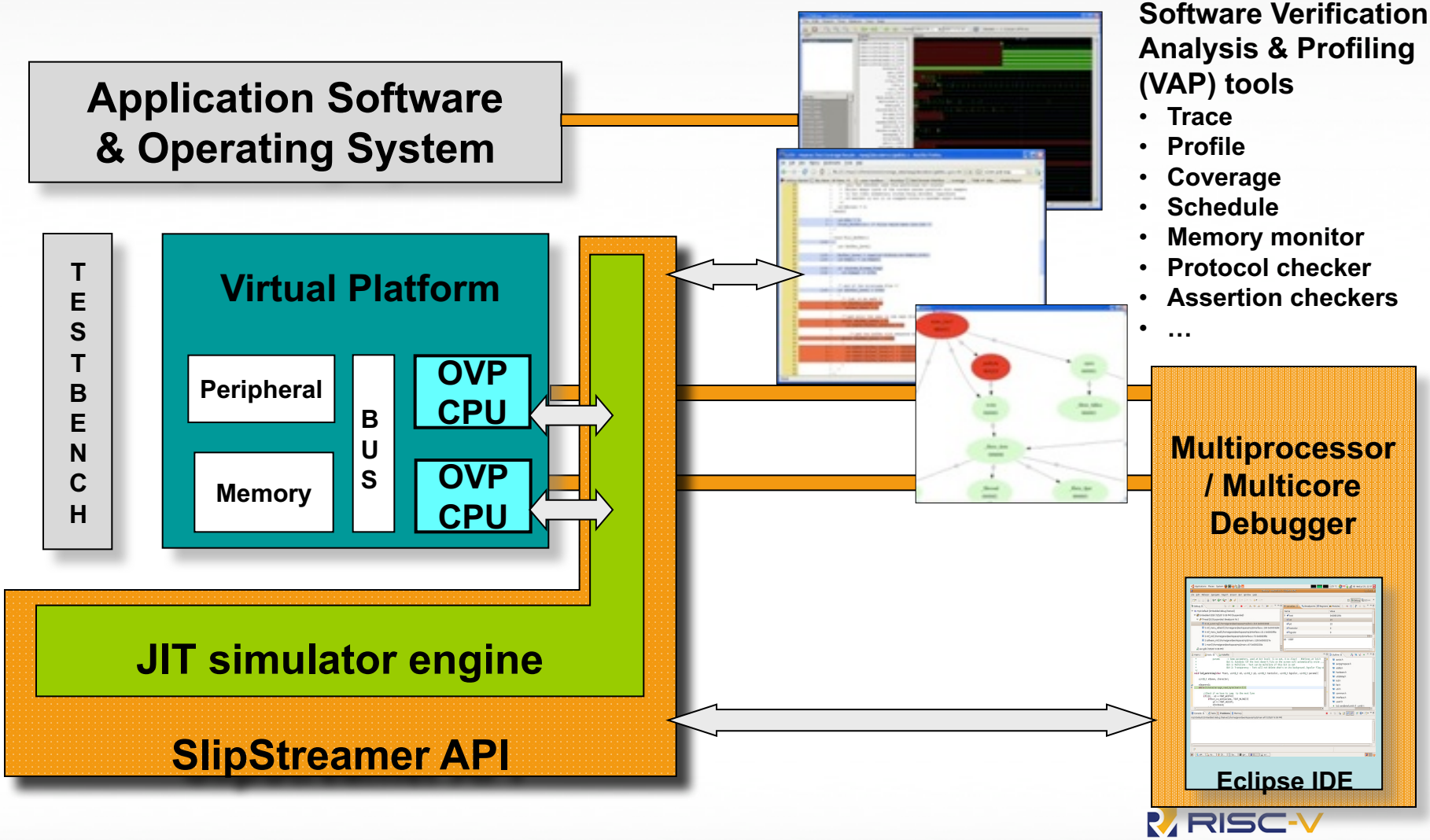
- Two parts to timing data
  - Cycle information – number of cycles for a given instruction, in context
  - Timing information – estimated time per cycle for a given silicon implementation
- Provided as a separately linked dynamic library
  - Enables processor designers to create a cycle approximate timing simulation without sharing any internal information

Agenda

- Why is timing estimation important for embedded systems?
- Current techniques for timing estimation
- Instruction accurate simulation
- Instruction Accurate + Estimation (IA+E)
- **Results**

Imperas Environment for Embedded Software Development, Debug & Test

imperas



Software Verification, Analysis & Profiling (VAP) tools

- Trace
- Profile
- Coverage
- Schedule
- Memory monitor
- Protocol checker
- Assertion checkers
- …

Characterization Example

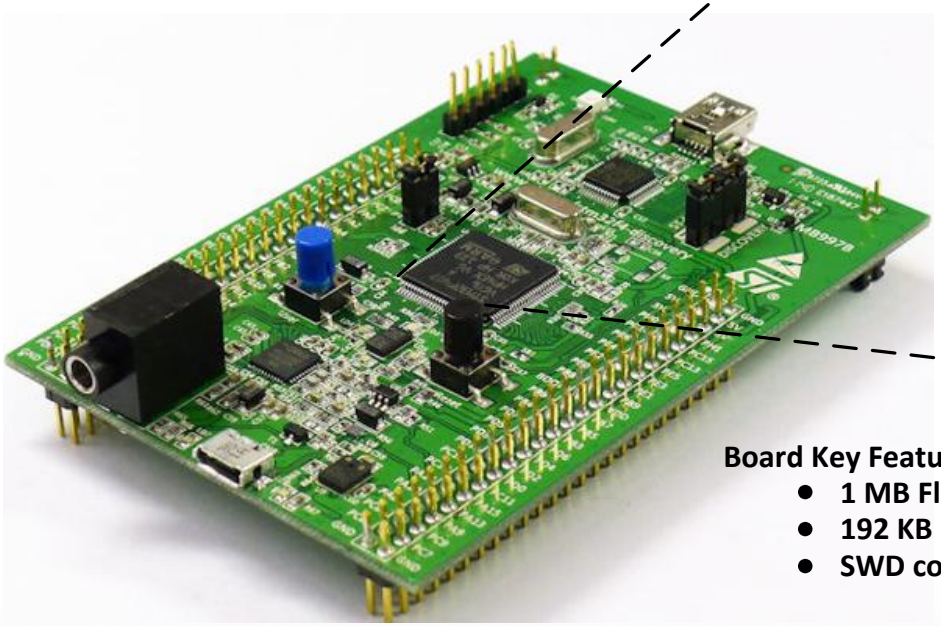
- Reference Board
  - STM32F4-Discovery board (ARM Cortex-M4F CPU)
- Cortex-M4F running FreeRTOS
  - both are highly used in high-performance embedded system design

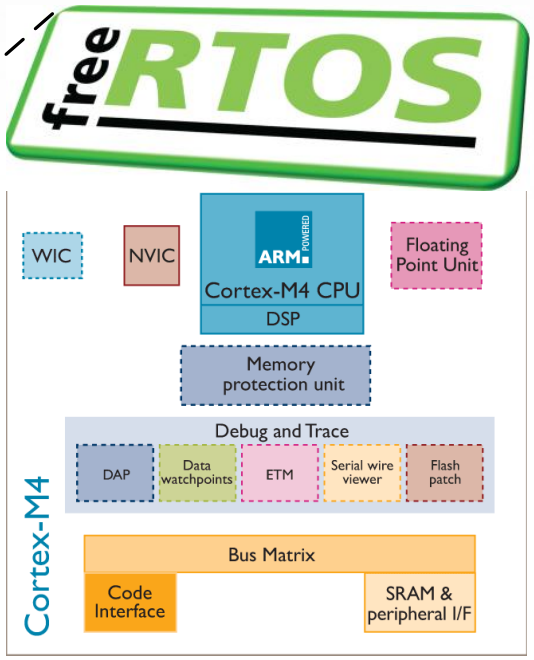
FreeRTOS Key Features:

- **kernel version V7.4.21**
- **POSIX wrapper (freertos_posix)**

ARM Cortex-M4 Key Features:

- **pipeline (3-stage + branch speculation)**
- **single precision floating point unit**
- **sleep modes for low power consumption**



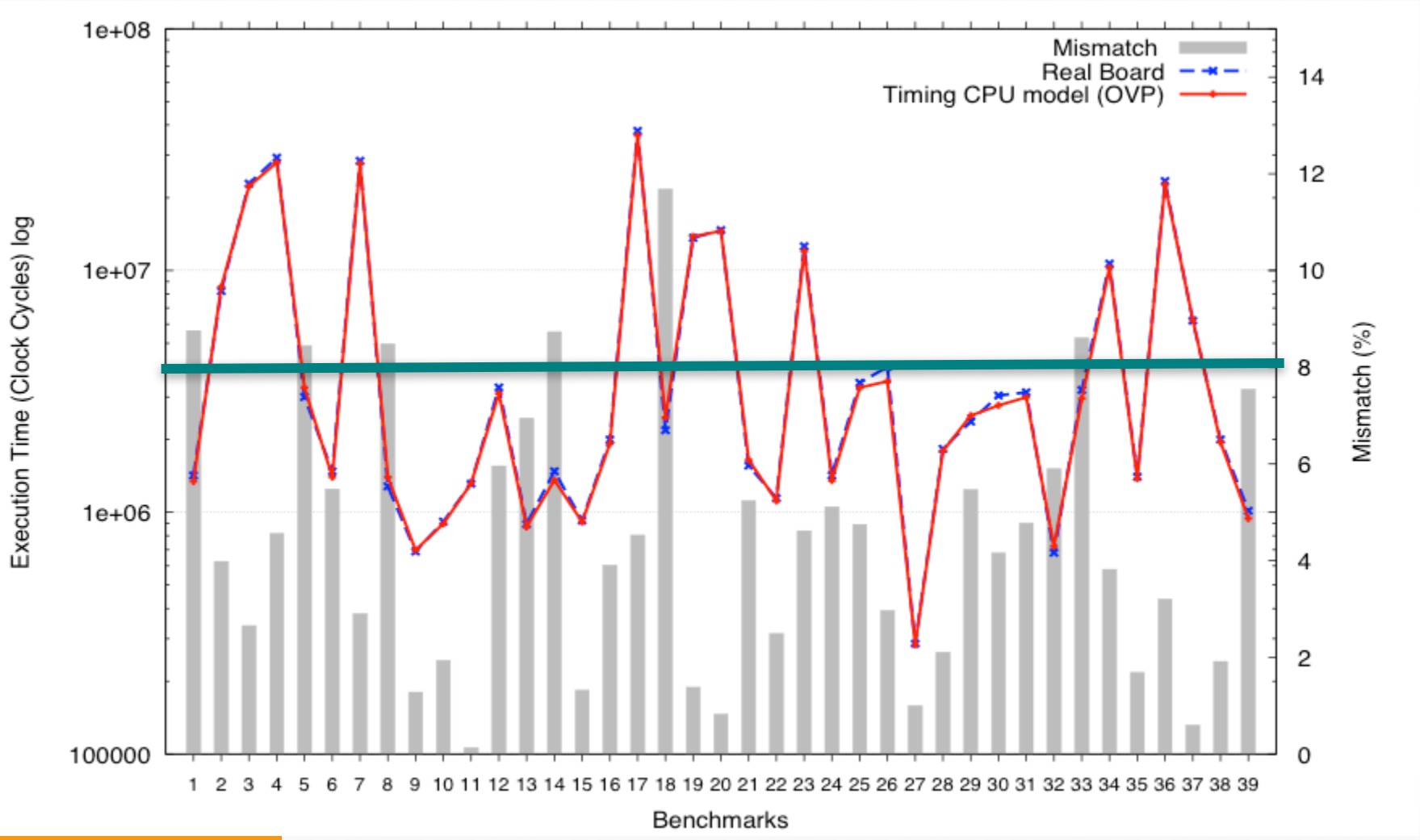


Board Key Features:

- **1 MB Flash**
- **192 KB RAM**
- **SWD connector for programming and debugging**

Accuracy Evaluation vs Board

- Experimental Runs
  - Cortex-M4F running FreeRTOS
  - WCET and other benchmarks



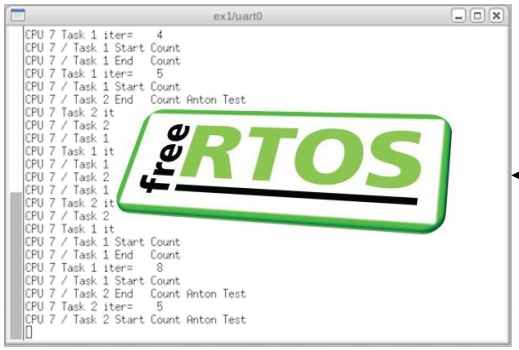
Performance

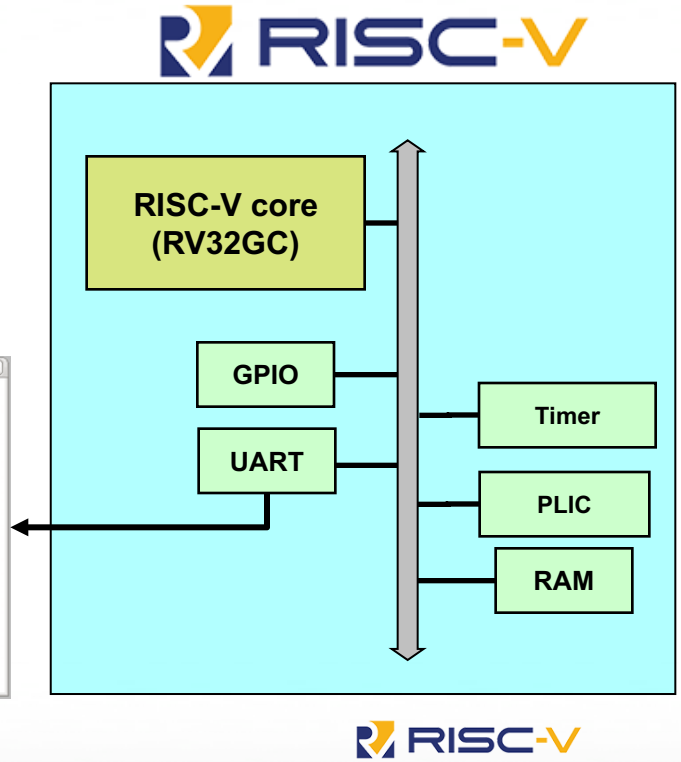
- For Cortex-M4F CPU estimator vs. Board
- Worst case error <13%
- Average error across all benchmarks run <5%
- Most errors are <8%
- Simulation speeds with timing up to 500 MIPS

- Caveat
  - Performance and accuracy are application dependent

Example RISC-V Platform

- RISC-V RV32 processor model
- Various peripheral models
- Several benchmark applications run, with different compiler optimizations



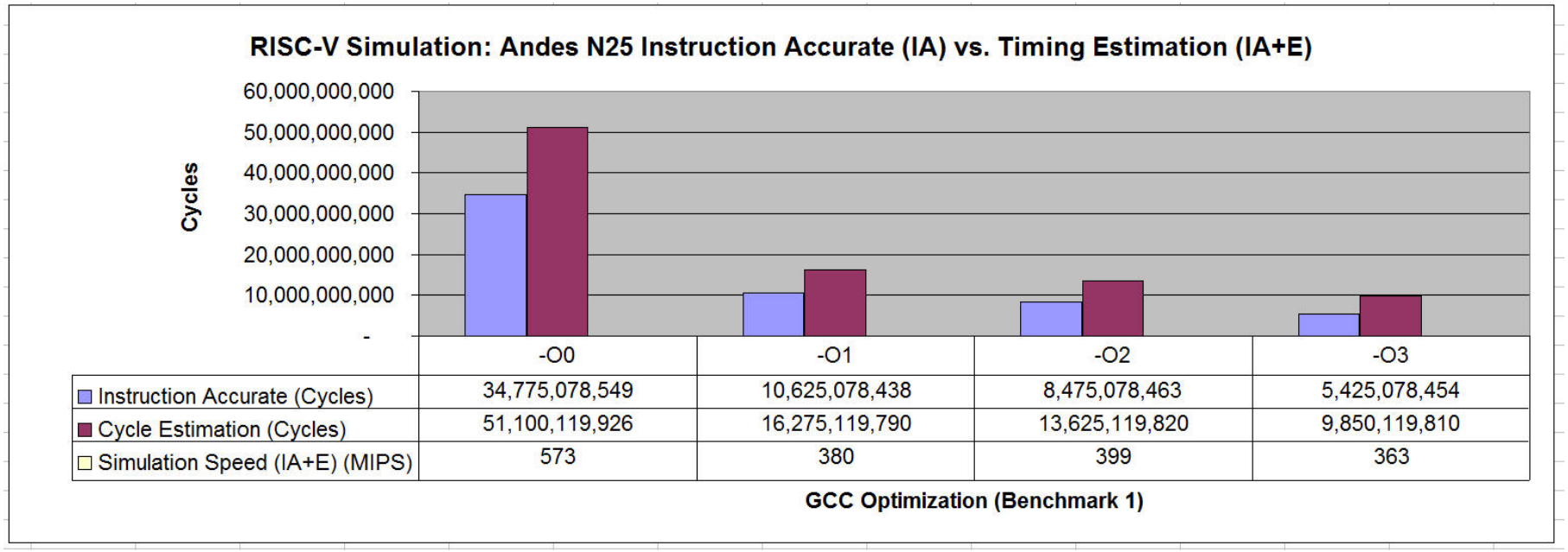


RISC-V Experiments

- Timing estimation using IA+E for RISC-V 32 bit
  - Andes Technology N25
  - Microsemi Mi-V RV32IMA
  - SiFive E31
- Different benchmarks used for each processor
  - Do not want to compare processor performance
- Only cycle information results are presented, so that we were not providing misleading timing estimation data on different processors

Andes N25 Results

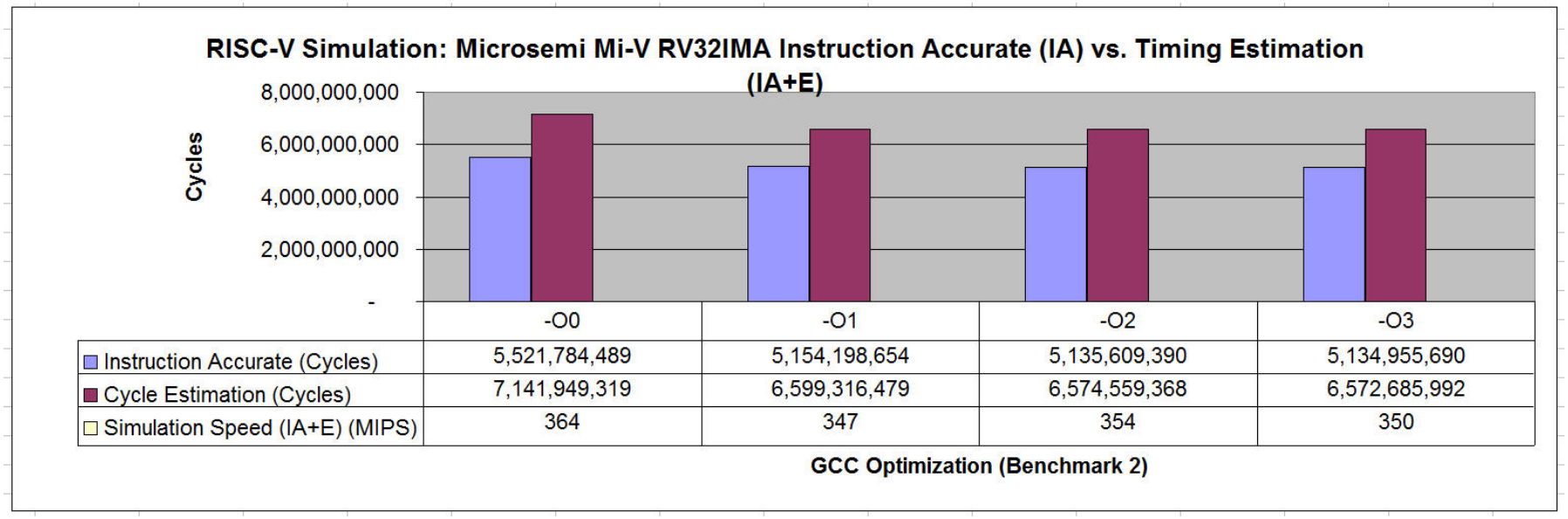
Comparing:
IA = 1 cycle per instruction
IA+E = estimated cycles per instruction



	-O0	-O1	-O2	-O3
Instruction Accurate (Cycles)	34,775,078,549	10,625,078,438	8,475,078,463	5,425,078,454
Cycle Estimation (Cycles)	51,100,119,926	16,275,119,790	13,625,119,820	9,850,119,810
Simulation Speed (IA+E) (MIPS)	573	380	399	363

Microsemi Mi-V RV32IMA Results

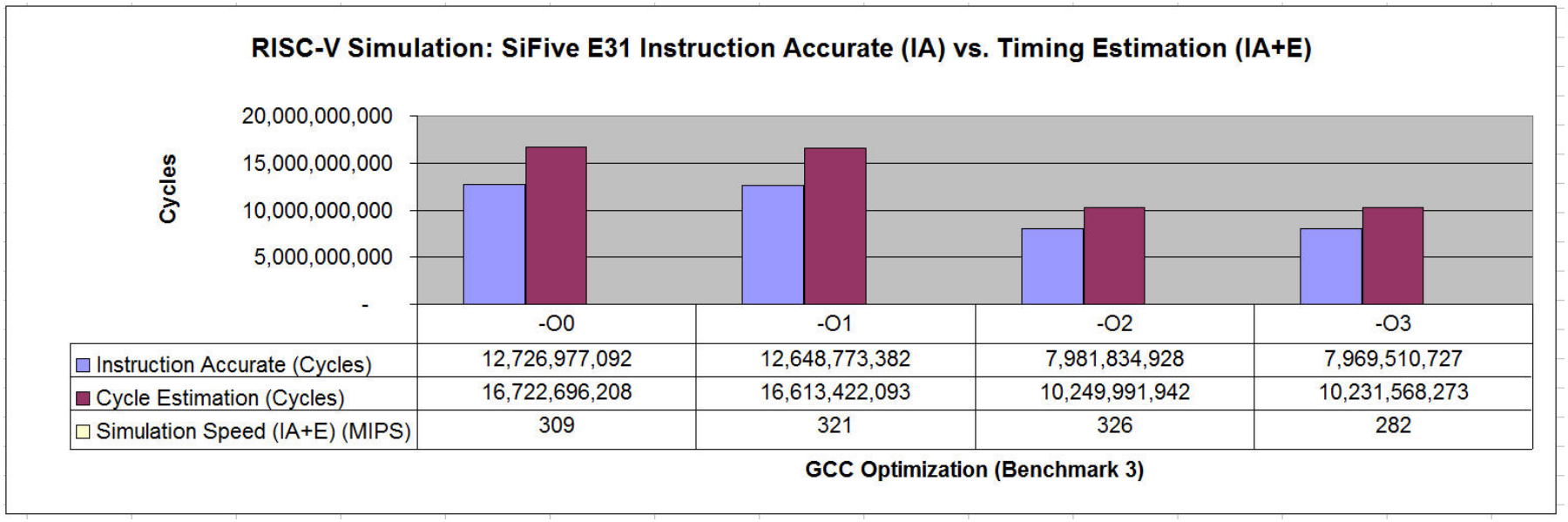
Comparing:
IA = 1 cycle per instruction
IA+E = estimated cycles per instruction



	-O0	-O1	-O2	-O3
Instruction Accurate (Cycles)	5,521,784,489	5,154,198,654	5,135,609,390	5,134,955,690
Cycle Estimation (Cycles)	7,141,949,319	6,599,316,479	6,574,559,368	6,572,685,992
Simulation Speed (IA+E) (MIPS)	364	347	354	350

SiFive E31 Results

Comparing:
IA = 1 cycle per instruction
IA+E = estimated cycles per instruction



	-O0	-O1	-O2	-O3
Instruction Accurate (Cycles)	12,726,977,092	12,648,773,382	7,981,834,928	7,969,510,727
Cycle Estimation (Cycles)	16,722,696,208	16,613,422,093	10,249,991,942	10,231,568,273
Simulation Speed (IA+E) (MIPS)	309	321	326	282

Summary & Conclusions

- IA+E technique shows excellent results for speed and minimal overhead, with acceptable accuracy
  - Very fast (up to 500 MIPS)
  - Simple timing estimation, or simulation time stretching
- Limitations
  - In-Order deterministic processors only
- Further work
  - Extend to more complex processors: cache, multi-core, out-of-order execution
  - Apply this technique to power estimation

Thank You!

- See Imperas at the RISC-V Foundation booth, Hall 3A-419

- Stay for the next RISC-V presentation, after the break:

  Securing RISC-V Machines Dynamically with Hardware-Enforced Metadata Policies, Steve Milburn, Dover Microsystems