Getting started with RISC-V custom instructions

Embedded World ‘23

Jon Taylor

15 March 2023

embeddedworld2023

Exhibition&Conference

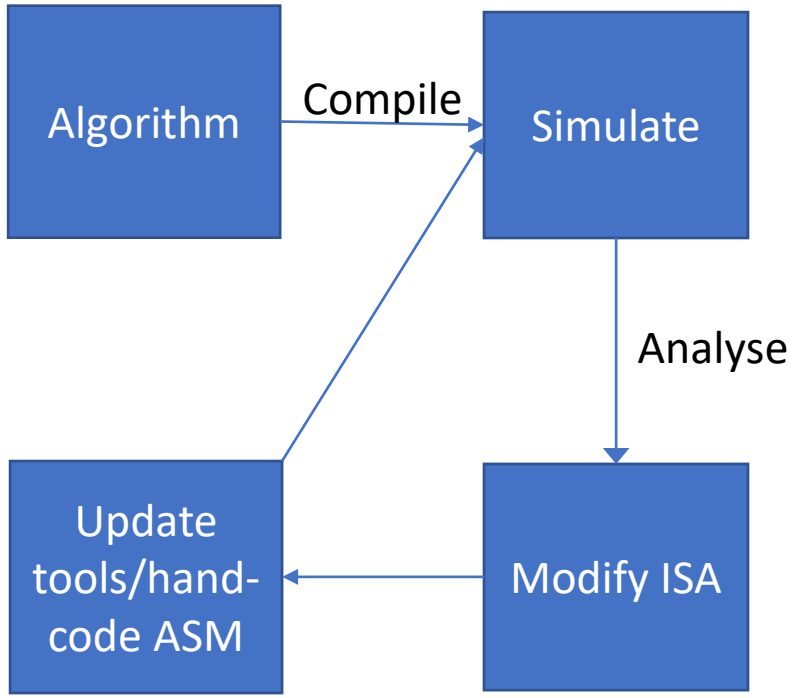
... it´s a smarter world

Why custom instructions?

- You already know your application inside out and know there are critical loops that could be improved
  - Allow you to run at a lower frequency
- You're migrating from other architectures and are missing a key instruction
- Additional functionality to simplify overall design (ie replacing MCU + DSP with customised MCU)

How to start customising

- Models let you explore quickly
  - Much faster to develop than RTL
  - Better profiling information available
  - Easier to debug software

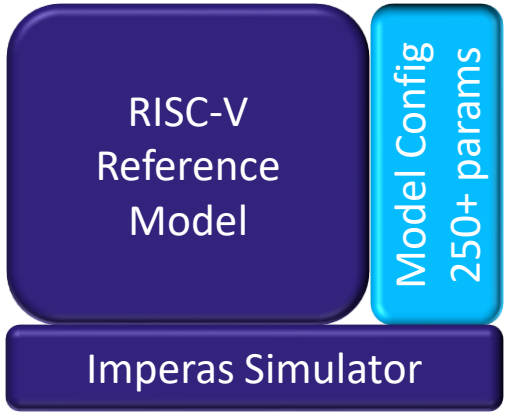


RISC-V Model Requirements

- Model the ISA, including all versions of the ratified spec, and stable unratified extensions
- Model other behavioral components, e.g. interrupt controllers
- Easily update and configure the model(s) for the next project
- User-extendable for custom instructions, registers, …
- Model actual processor IP, e.g. Andes, SiFive, Codasip, MIPS, OpenHW, Mi-V, …
- Well-defined test process including coverage metrics
- Interface to other simulators, e.g. SystemVerilog, SystemC, Imperas virtual platform simulators
- Interface to software debug tools, e.g. GDB/Eclipse, Imperas MPD
- Interface to software analysis tools including access to processor internal state, etc.
- Interface to architecture exploration tools including extensibility to timing estimation

- Most RISC-V ISSs can meet one or two of these requirements
- Imperas models and simulators were built to satisfy these requirements, and matured through usage on non-RISC-V ISAs over the last 12+ years

Imperas is the Processor Model

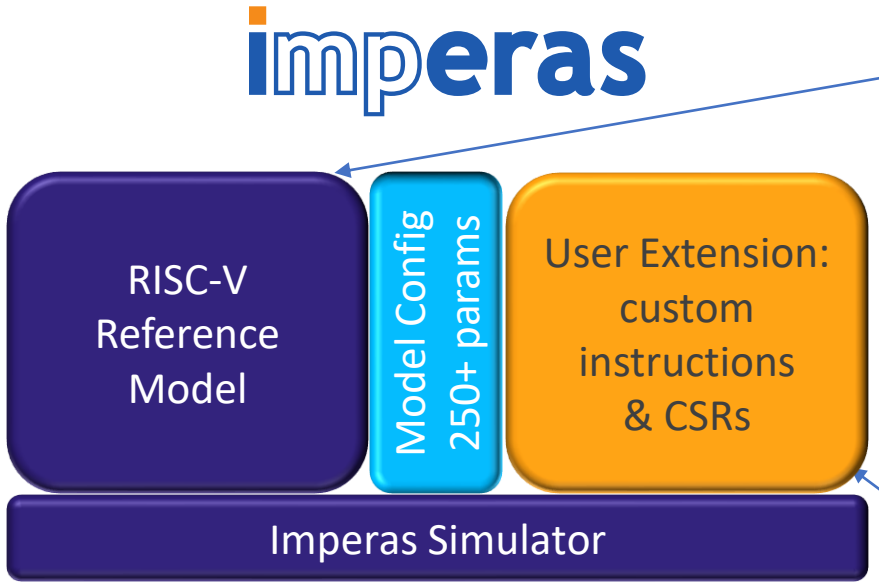


http://www.imperas.com/riscv

- Imperas provides full RISC-V Specification envelope model
- Industrial quality model /simulator of RISC-V processors for use in verification, software development and architecture exploration
- Complete, fully functional, configurable model / simulator
  - All 32bit and 64bit features of ratified User and Privilege RISC-V specs
  - Vector extension, versions 0.7.1, 0.8, 0.9, 1.0
  - Bit Manipulation extension, versions 0.91, 0.92. 0.93, 1.0.0
  - Hypervisor version 0.6.1
  - K-Crypto Scalar version 0.7.1, 1.0.0
  - Debug versions 0.13.2, 0.14, 1.0.0

"The donation of a robust, commercial-quality simulator – riscvOVPsim – will enable our customers to adopt RISC-V even faster. This is the level of close industry collaboration that will drive the successful adoption of RISC-V."
Yunsup Lee, co-founder and CTO, SiFive

Imperas Model Extensibility



- Separate source files and no duplication to ensure easy maintenance
- Imperas or user can develop the extension
- User extension source can be proprietary

Imperas develops and maintains base model

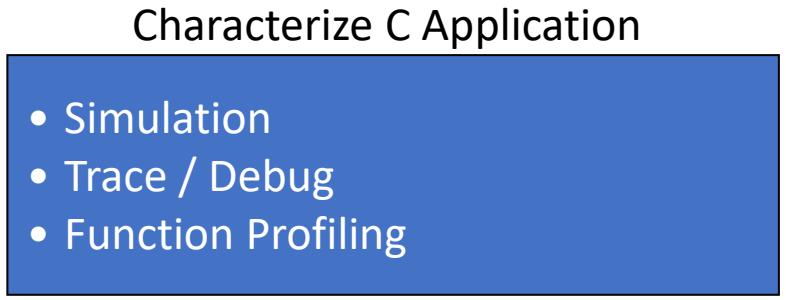
- Base model implements RISC-V specification in full
- Fully user configurable to select which ISA extensions
- Fully user configurable to select which version of each ISA extension
  - Updated very regularly as ISA extension specification versions change
- Fully user configurable for all RISC-V specification options
  - e.g. implemented optional CSRs, read only or read/write bits etc…

Imperas provides methodology to easily extend base model

- Templates to add new instructions
- Code fragment for adding functionality
- 100+ page user guide/reference manual with many examples
  - Includes example extended processor model

Imperas OVP model is architected for easy extension & maintenance

Flow to Add New Custom Instructions



- Custom instructions are added to optimize a specific application or set of applications within a domain
- Therefore, start by characterizing the application to be optimized
- Then add the custom instructions, evaluate, and iterate

Simulation of C Application

- Example: Character stream encoder, with ChaCha20 encryption algorithm (adding ChaCha20 specific custom instructions)
- Cross compiled C application targeting RV32IM
- Instruction Accurate (IA) simulation
  - Imperas simulator with configurable model of RISC-V specification selecting RV32IM

➢ Runs fast
  - Over 1 billion instructions per second (standard PC)

Function Profile C Application

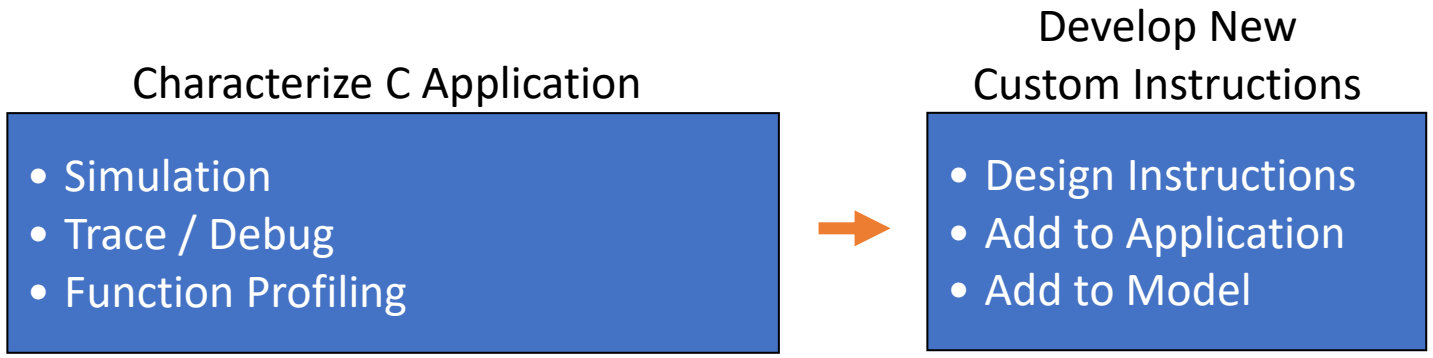
- Same C application
- Sampled profiling with call stack analysis

➢Shows proportion of time spent in each application function

➢21.35% spent in processLine

Name (location)	Arcs in	Samples In	Arcs out	From/To this	Percentage (%)
Platform: iss					
Processor: iss/cpu0					
Process: 0_None		1659204277			
_fread_r	598189652	596534872	1654780		35.95%
processLine	925852930	354236017	571616913		21.35%
qr4_c	150627639	150627639	0		9.08%
qr1_c	146083640	146083640	0		8.8%
qr2_c	137682652	137682652	0		8.3%
qr3_c	137222982	137222982	0		8.27%
__libc_init_array	0	135154865	1524049412		8.15%
__srefill_r	1654780	1024985	629795		0.06%
__sread	629637	321116	308521		0.02%
_read_r	308521	308521	0		0.02%
_fseeko_r	2706	2126	580		0.0%
_vfprintf_r	1874	764	1110		0.0%
__sfvwrite_r	848	752	96		0.0%
rewind	3267	561	2706		0.0%
_close_r	357	357	0		0.0%
_malloc_r	323	297	26		0.0%
__sseek	528	288	240		0.0%
_lseek_r	240	240	0		0.0%
__sfmoreglue	399	224	175		0.0%
_fclose_r	734	204	530		0.0%

Flow to Add New Custom Instructions



Add Custom Instructions to Model and Re-Simulate

- Use standard Open Virtual Platforms (OVP) instruction modeling APIs to add new instructions (and optional state) as ***new extension library***
- Compile and link model extension library
- Simulate standard model extended with new library

➢ Instruction count and simulated time have been reduced

```
//
// Create the RISCV decode table
//
static vmidDecodeTableP createDecodeTable(void) {

    vmidDecodeTableP table = vmidNewDecodeTable(RISCV_INSTR_BITS, RISCV_EIT_LAST);

    // R-Type instruction in custom-0 encoding space:
    // opcode [6:0]  = 00 010 11
    // funct3[14:12] = 0,1,2,3 (QR1-4)
    // funct7[31:25] = 0000000
    // rs1[19:15]
    // rs2[24:20]
    // rd[11:7]

    // handle custom instruction
    DECODE_ENTRY(0, CHACHA20QR1, "|0000000.........000.....0001011|");
    DECODE_ENTRY(0, CHACHA20QR2, "|0000000.........001.....0001011|");
    DECODE_ENTRY(0, CHACHA20QR3, "|0000000.........010.....0001011|");
    DECODE_ENTRY(0, CHACHA20QR4, "|0000000.........011.....0001011|");

    return table;
}
```

```
//
// Emit code implementing exchange instruction
//
static void emitChaCha20(
    vmiProcessorP processor,
    vmiosObjectP  object,
    Uns32         instruction,
    Uns32         rotl
) {

    // extract instruction fields
    Uns32 rd  = RD(instruction);
    Uns32 rs1 = RS1(instruction);
    Uns32 rs2 = RS2(instruction);

    vmiReg reg_rs1 = vmimtGetExtReg(processor,  &object->rs1);
    vmiReg reg_rs2 = vmimtGetExtReg(processor,  &object->rs2);
    vmiReg reg_tmp = vmimtGetExtTemp(processor, &object->tmp);

    vmimtGetR(processor, RISCV_REG_BITS, reg_rs1, object->riscvRegs[rs1]);
    vmimtGetR(processor, RISCV_REG_BITS, reg_rs2, object->riscvRegs[rs2]);
    vmimtBinopRRR(32, vmi_XOR, reg_tmp, reg_rs1, reg_rs2, 0);
    vmimtBinopRC(32, vmi_ROL, reg_tmp, rotl, 0);

    vmimtSetR(processor, RISCV_REG_BITS, object->riscvRegs[rd], reg_tmp);
}
```

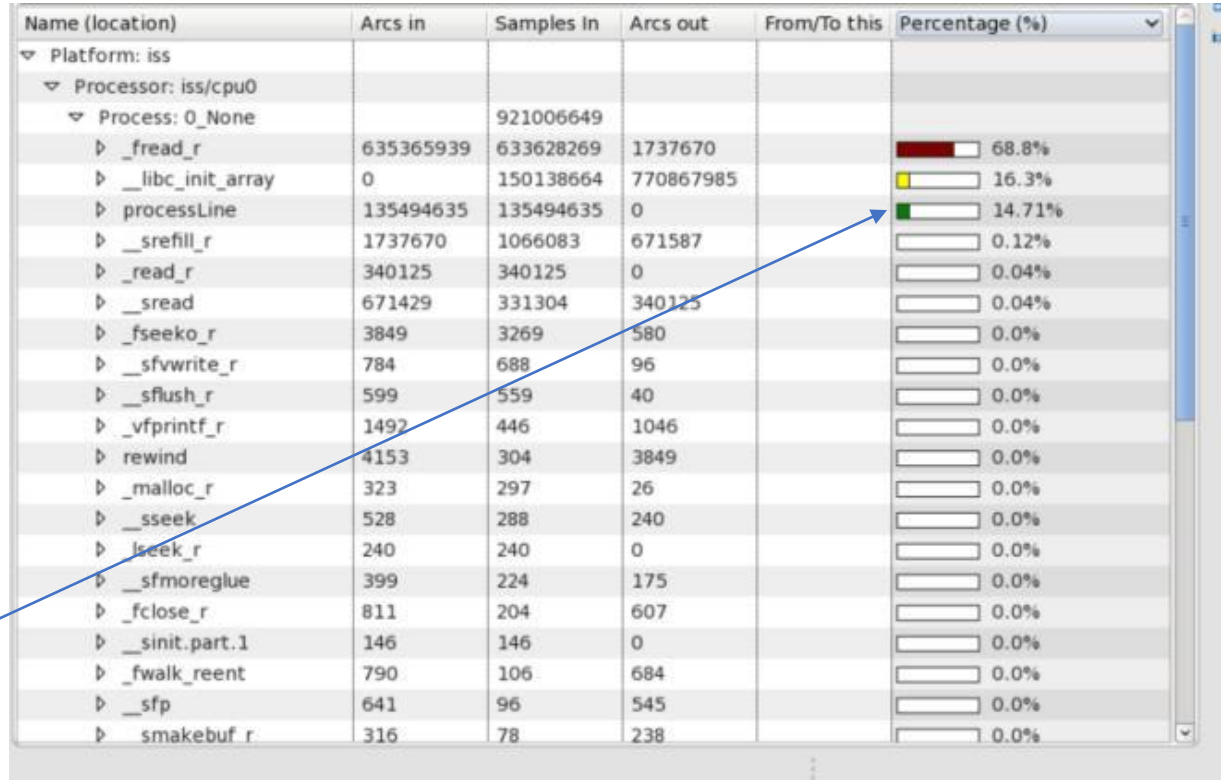
```
CpuManagerMulti (32-Bit) v99999999 Open Virtual Platform simulator from www.IMPERAS.com.
Copyright (c) 2005-2018 Imperas Software Ltd.  Contains Imperas Proprietary Information.
Licensed Software, All Rights Reserved.
Visit www.IMPERAS.com for multicore debug, verification and analysis solutions.

CpuManagerMulti started: Thu Aug 23 11:41:32 2018

Info (OP_LPR) Processor  iss/cpu0                              $IMPERAS_VLNV/riscv.ovpworld.
Info (OR_OF) Target 'iss/cpu0' has object file read from 'application/test_custom.RISCV32.elf'
Info (OR_PH) Program Headers:
Info (OR_PH) Type           Offset     VirtAddr   PhysAddr   FileSiz    MemSiz     Flags Align
Info (OR_PD) LOAD           0x00000000 0x00010000 0x00010000 0x00017270 0x00017270 R-E   1000
Info (OR_PD) LOAD           0x00017270 0x00028270 0x00028270 0x000009c0 0x00000a24 RW-   1000
Info (OR_OF) Target 'iss/cpu0' has object file read from 'application/exception.RISCV32.elf'
Info (OR_PH) Program Headers:
Info (OR_PH) Type           Offset     VirtAddr   PhysAddr   FileSiz    MemSiz     Flags Align
Info (OR_PD) LOAD           0x00001000 0x00000000 0x00000000 0x0000000c 0x0000000c R-E   1000
Info (OP_PEX) Extension  iss/cpu0/riscv32Newlib                $IMPERAS_VLNV/riscv.ovpworld.org/semihosting/riscv32Newlib/1.0/model
Info (OP_PEX) Extension  iss/cpu0/exInst                       instructionExtensionLib
RES = 84772366
Info
Info ------------------------------------------------------
Info CPU 'iss/cpu0' STATISTICS
Info   Type              : riscv (RV32IM)
Info   Nominal MIPS      : 100
Info   Final program counter : 0x100ac
Info   Simulated instructions: 677,012,570
Info   Simulated MIPS    : 1301.9
Info ------------------------------------------------------
Info
Info ------------------------------------------------------
Info SIMULATION TIME STATISTICS
Info   Simulated time    : 6.77 seconds
Info   User time         : 0.50 seconds
Info   System time       : 0.02 seconds
Info   Elapsed time      : 0.53 seconds
Info   Real time ratio   : 12.81x faster
Info ------------------------------------------------------

CpuManagerMulti finished: Thu Aug 23 11:41:33 2018
```

Function Profile Application Using Custom Instructions

- IA simulation + custom instructions with sampled profiling

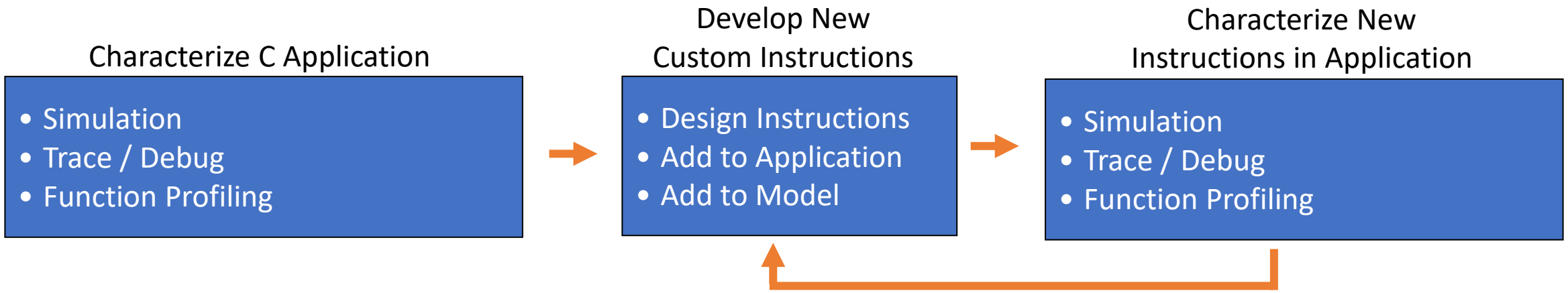
➢Shows where slowest function is
  - Now much faster…

➢Shows benefits of using custom instructions
  - ➢processLine was 21.35% now 14.71%

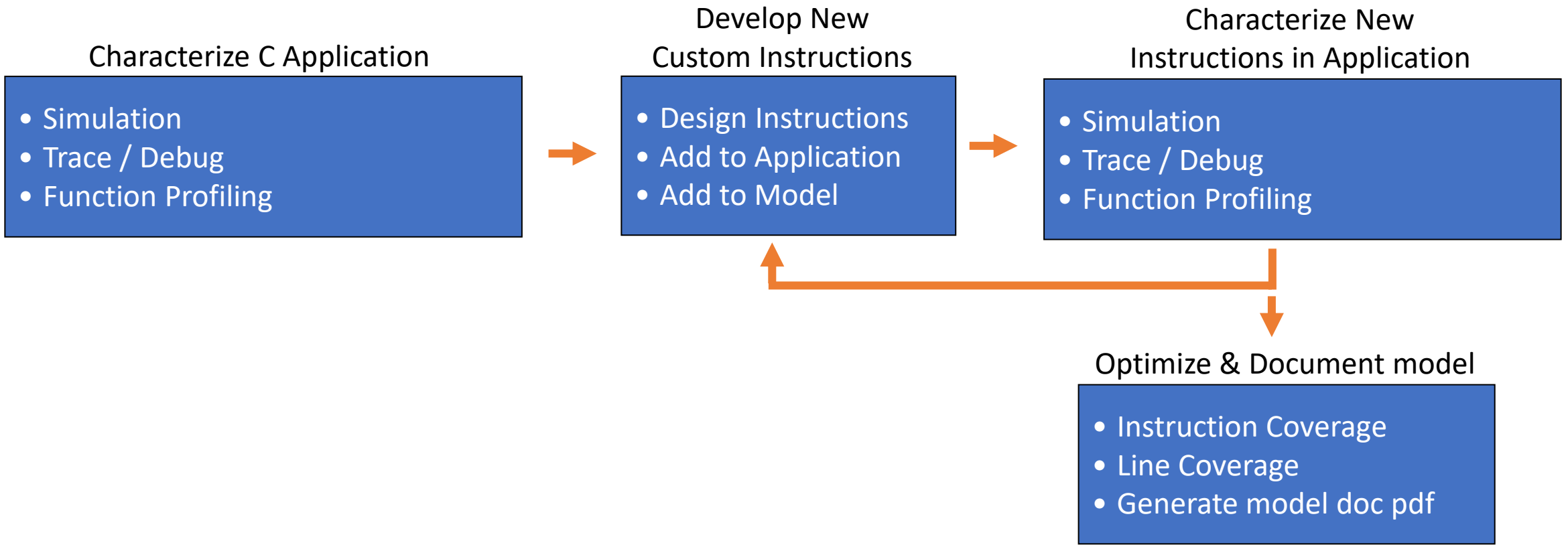


Name (location)	Arcs in	Samples in	Arcs out	From/To this	Percentage (%)
Platform: iss					
Processor: iss/cpu0					
Process: 0_None		921006649			
_fread_r	635365939	633628269	1737670		68.8%
__libc_init_array	0	150138664	770867985		16.3%
processLine	135494635	135494635	0		14.71%
__srefill_r	1737670	1066083	671587		0.12%
_read_r	340125	340125	0		0.04%
__sread	671429	331304	340125		0.04%
_fseeko_r	3849	3269	580		0.0%
__sfvwrite_r	784	688	96		0.0%
__sflush_r	599	559	40		0.0%
_vfprintf_r	1492	446	1046		0.0%
rewind	4153	304	3849		0.0%
_malloc_r	323	297	26		0.0%
__sseek	528	288	240		0.0%
_lseek_r	240	240	0		0.0%
__sfmoreglue	399	224	175		0.0%
_fclose_r	811	204	607		0.0%
__sinit.part.1	146	146	0		0.0%
_fwalk_reent	790	106	684		0.0%
__sfp	641	96	545		0.0%
__smakebuf_r	316	78	238		0.0%

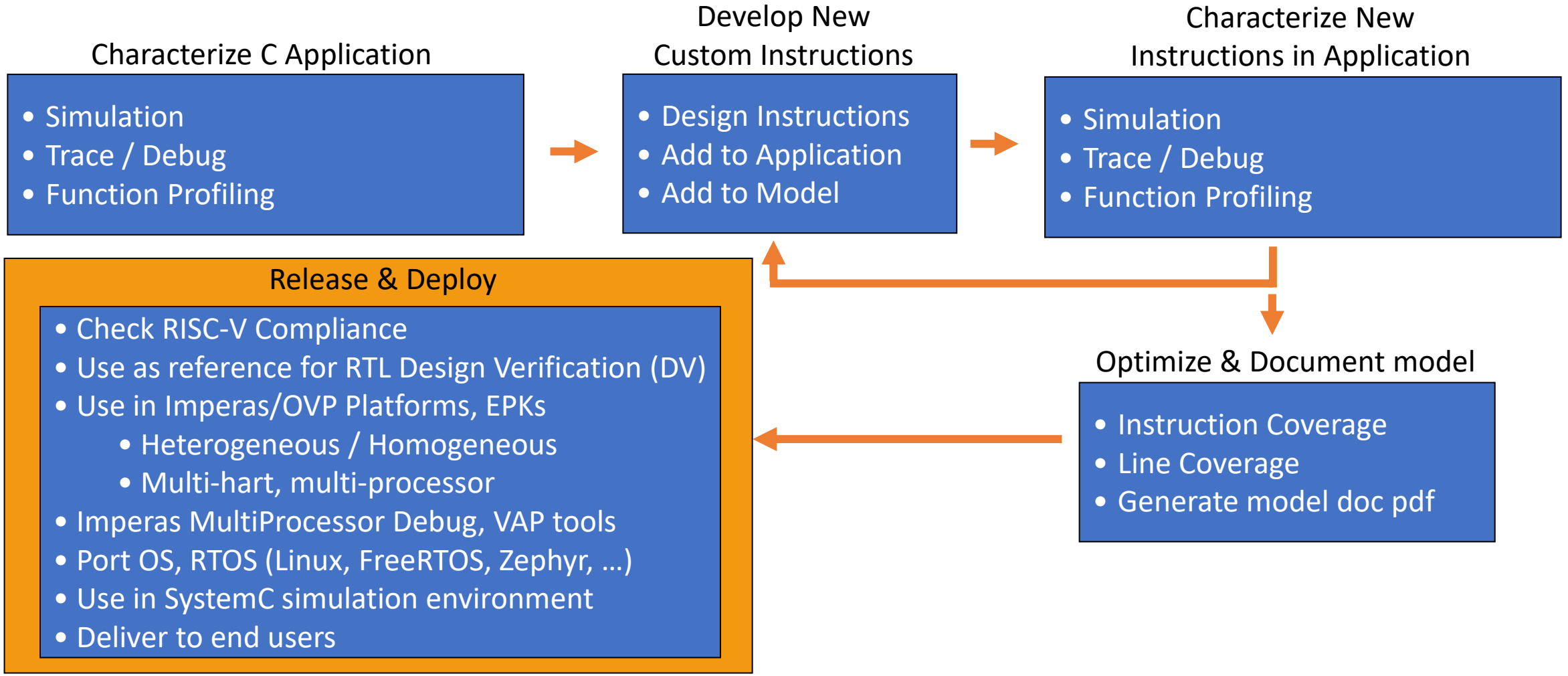
Flow to Add New Custom Instructions



Flow to Add New Custom Instructions



Flow to Add New Custom Instructions

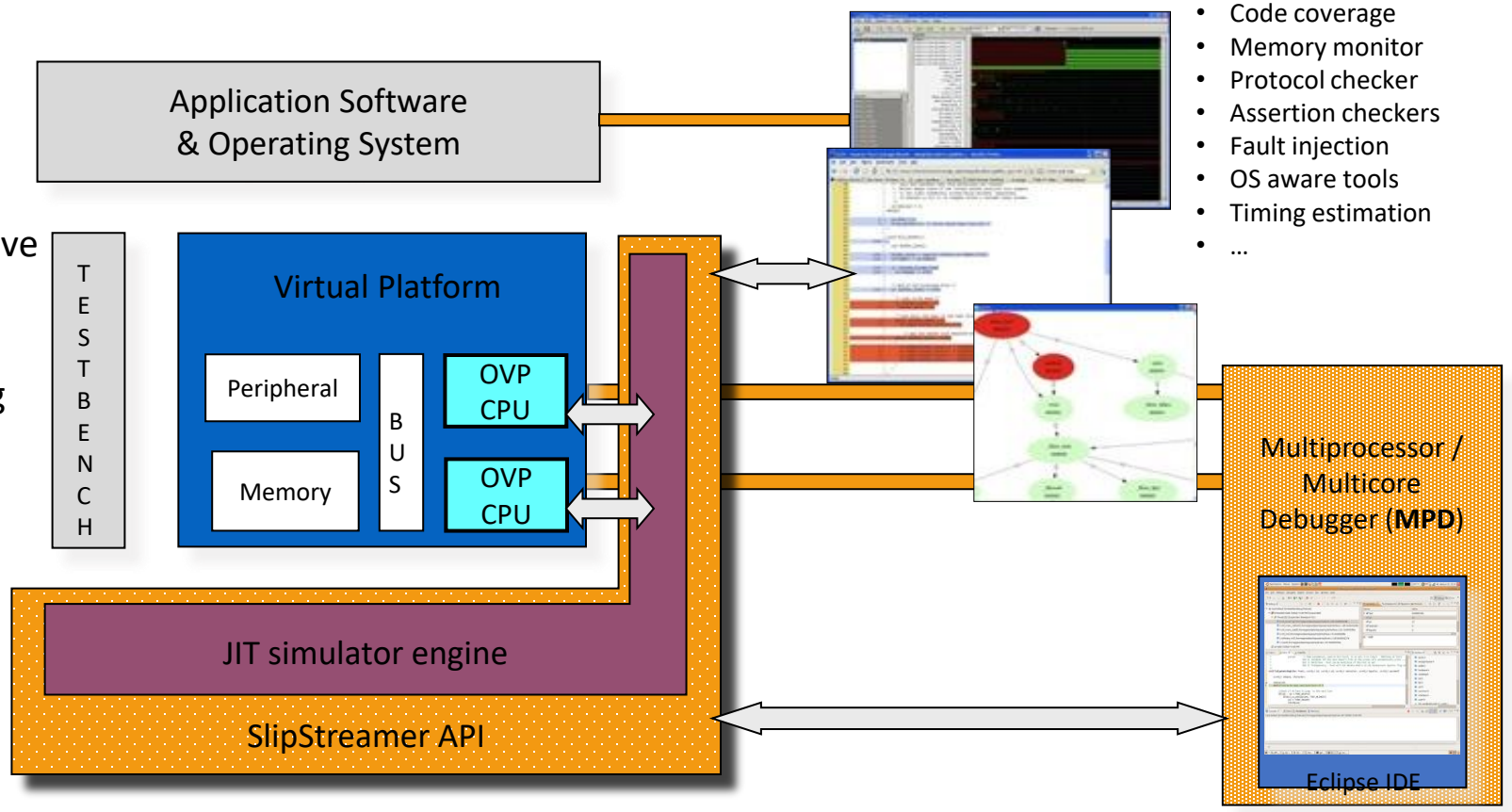


Environment for Software Development, Architecture Analysis

imperas

Key technologies/differentiators:

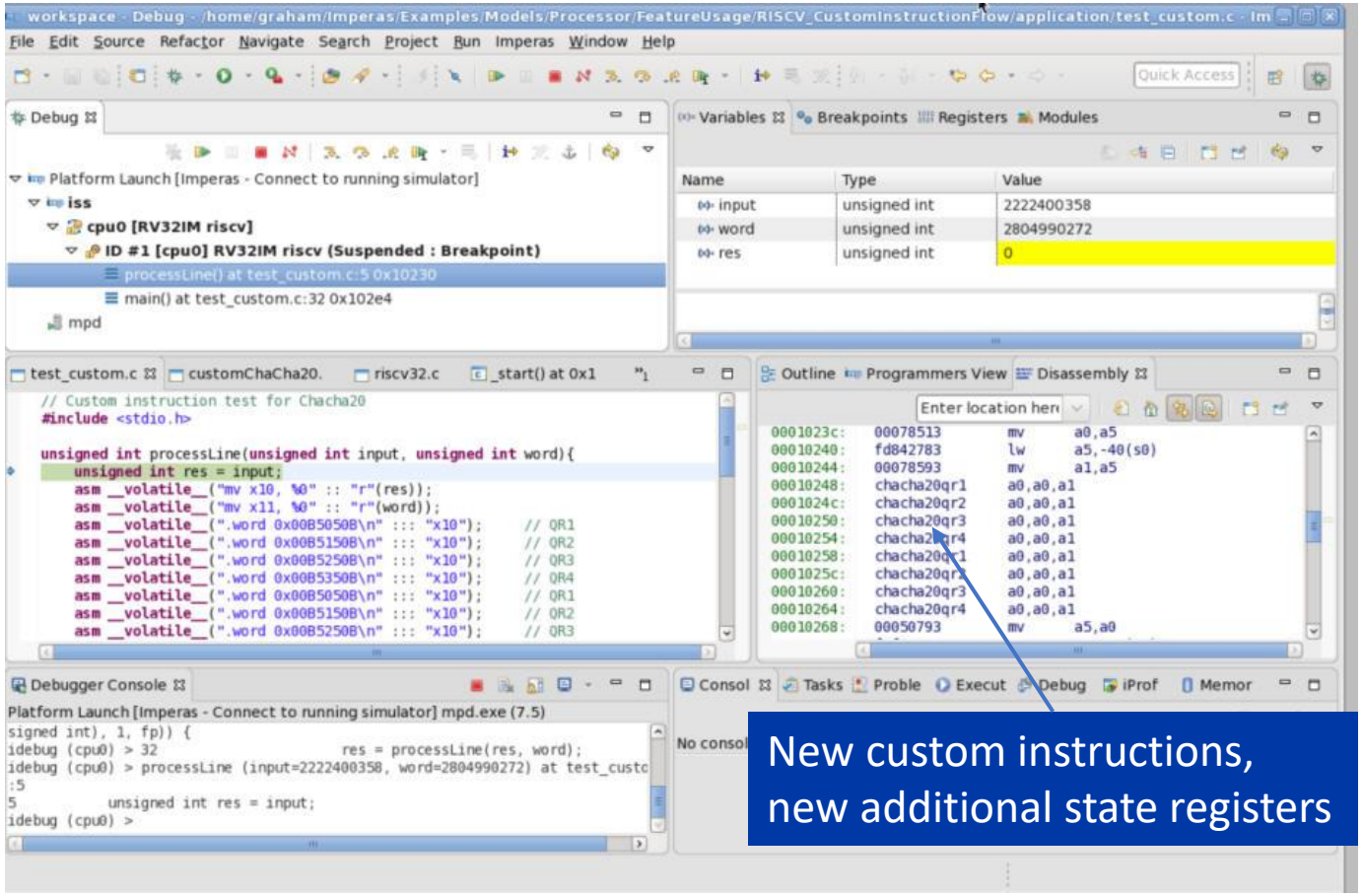
- OVP Fast Processor Models
  - Most models
  - Highest quality
- Simulator engine
  - Highest performance
  - SlipStreamer API for non-intrusive tools
- Tools
  - MPD for platform-centric debug
  - VAP tools for comprehensive software analysis



Software Verification, Analysis & Profiling (**VAP**) tools

- Trace (instruction, function, variable, …)
- Profile
- Code coverage
- Memory monitor
- Protocol checker
- Assertion checkers
- Fault injection
- OS aware tools
- Timing estimation
- …

Software Debug and Analysis Tools Automatically Work With the Custom Instructions



Thank you

Jon Taylor

JonT@imperas.com