2024
DESIGN AND VERIFICATION™
DVCON
CONFERENCE AND EXHIBITION
UNITED STATES
SAN JOSE, CA, USA
MARCH 4-7, 2024

RISC-V Testing
Status and current state of the art

Jon Taylor, Synopsys

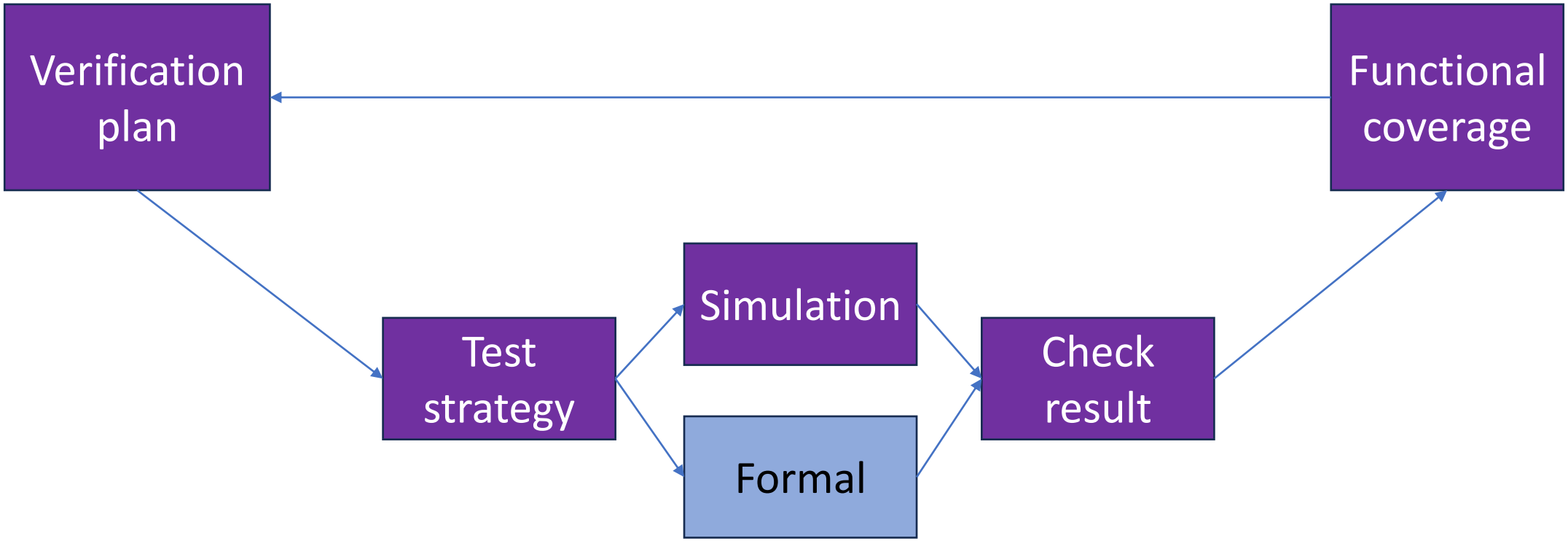
SYNOPSYS®
Silicon to Software™

accellera
SYSTEMS INITIATIVE

Agenda

- Introduction
- Test validation
- Test sources
- Functional coverage
- Microarchitectural testing and benchmarks
- Future work and conclusions

Introduction



Test validation methodologies

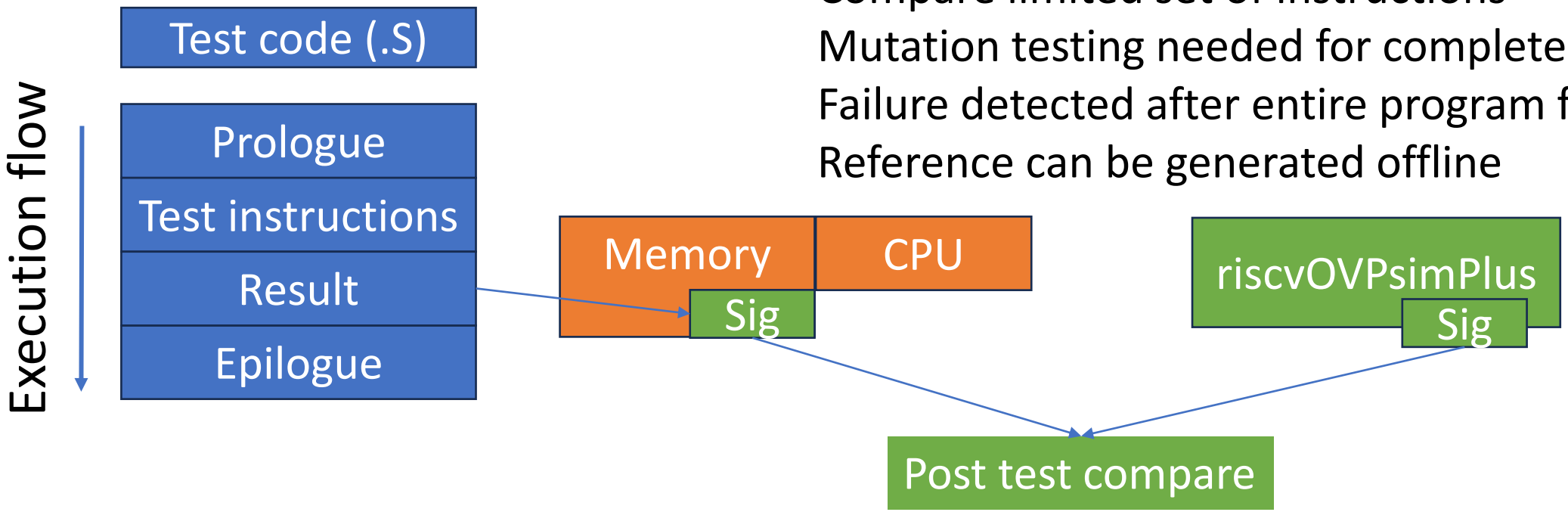
- Signature compare
- Trace compare
- Synchronous lockstep compare
- Asynchronous continuous compare

- All require a golden reference model

RISC-V Reference models

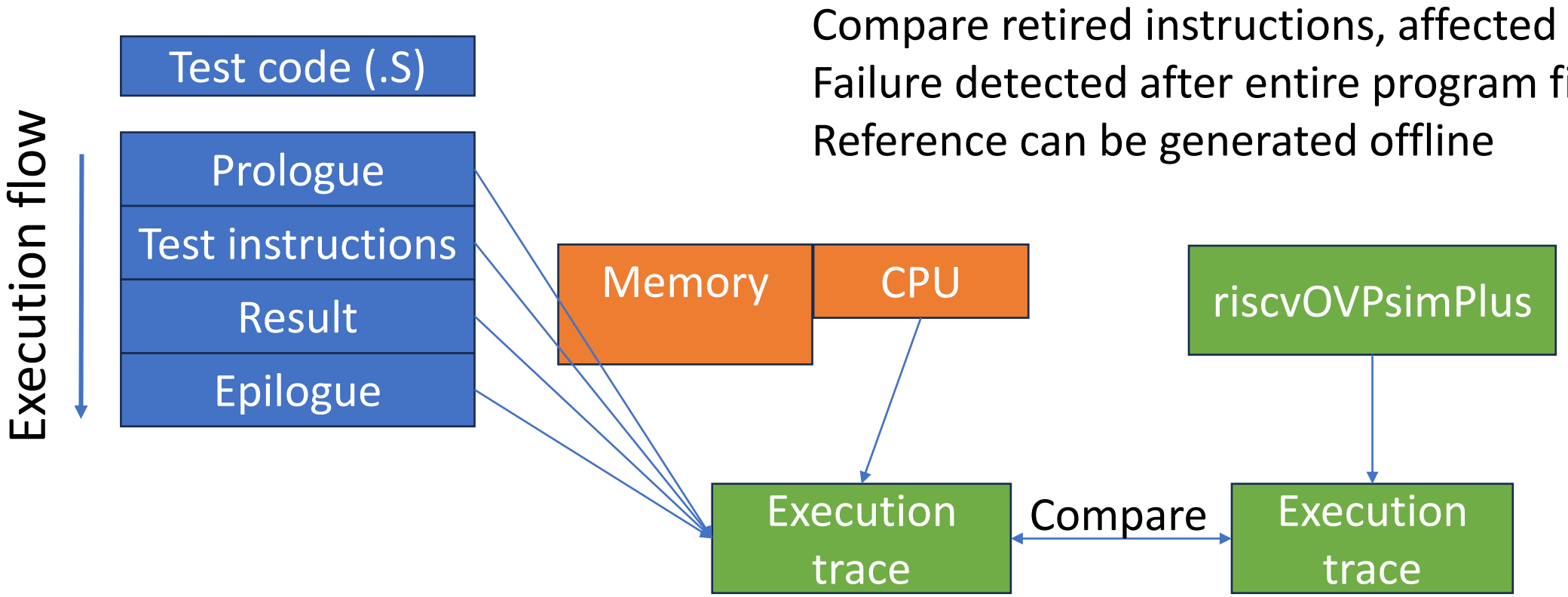
Simulator	QEMU	Spike	SAIL	ImperasFPM
License	GPL	UC permissive	BSD permissive	Commercial
Feature completeness	All ISA features	Latest spec versions only	No PMA, no hyp, no vectors	All features, all versions
Signature compare	✗	✓	✓	✓
Trace compare	✗	✓	✓	✓
Sync lockstep	✗	✓	✗	✓
Async compare	✗	✗	✗	✓
Supports Hardware Assisted Verification	✗	✗	✗	✓
Supports Virtual Platforms and Hybrid simulation	✓	✗	✗	✓

Signature compare



Compare limited set of instructions
Mutation testing needed for completeness
Failure detected after entire program finishes
Reference can be generated offline

Trace compare

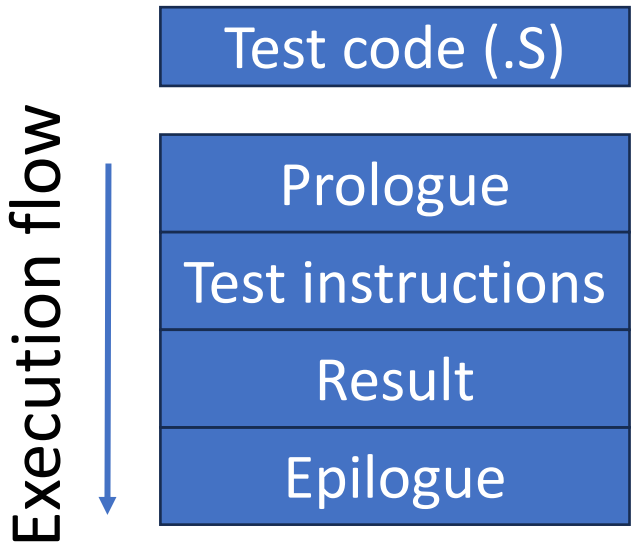


Compare retired instructions, affected registers
Failure detected after entire program finishes
Reference can be generated offline

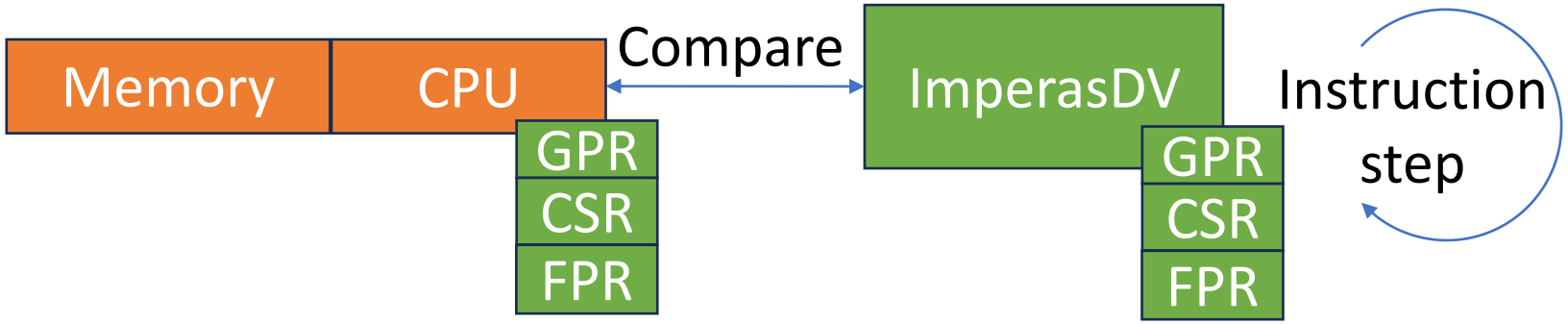
Comparison isn’t trivial

riscvOVPsimPlus	Spike
Info 1: 'riscvOVPsim/cpu', 0x0000000080000000(_start): Machine f14022f3 csrr t0,mhartid Info 2: 'riscvOVPsim/cpu', 0x0000000080000004(_start+4): Machine 4301 li t1,0 Info 3: 'riscvOVPsim/cpu', 0x0000000080000006(_start+6): Machine 00628263 beq t0,t1,8000000a Info 4: 'riscvOVPsim/cpu', 0x000000008000000a(_start+a): Machine 00000417 auipc s0,0x0 Info s0 0000000000000000 -> 000000008000000a Info 5: 'riscvOVPsim/cpu', 0x000000008000000e(_start+e): Machine 00c40413 addi s0,s0,12 Info s0 000000008000000a -> 0000000080000016 Info 6: 'riscvOVPsim/cpu', 0x0000000080000012(_start+12): Machine 00040067 jr s0	core 0: 0x0000000080000000 (0xf14022f3) csrr t0, mhartid core 0: 3 0x0000000080000000 (0xf14022f3) x5 0x0000000000000000 core 0: 0x0000000080000004 (0x00004301) c.li t1, 0 core 0: 3 0x0000000080000004 (0x4301) x6 0x0000000000000000 core 0: 0x0000000080000006 (0x00628263) beq t0, t1, pc + 4 core 0: 3 0x0000000080000006 (0x00628263) core 0: 0x000000008000000a (0x00000417) auipc s0, 0x0 core 0: 3 0x000000008000000a (0x00000417) x8 0x000000008000000a core 0: 0x000000008000000e (0x00c40413) addi s0, s0, 12 core 0: 3 0x000000008000000e (0x00c40413) x8 0x0000000080000016 core 0: 0x0000000080000012 (0x00040067) jr s0 core 0: 3 0x0000000080000012 (0x00040067)

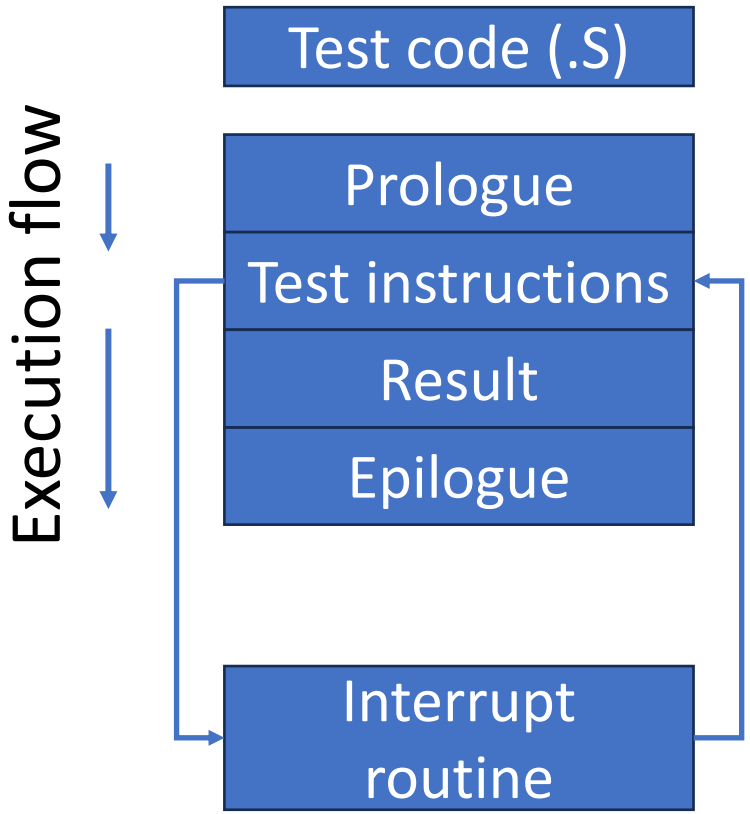
Synchronous lockstep compare



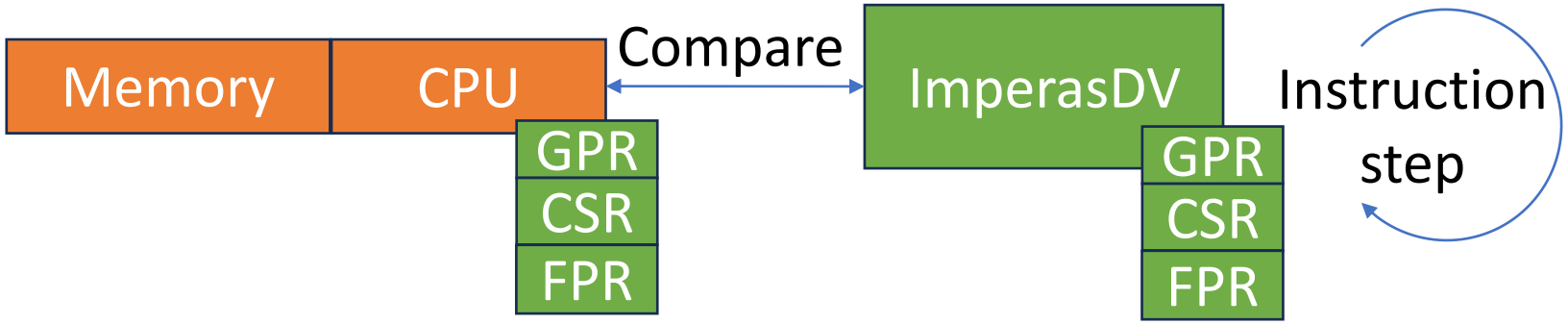
Compare retired instruction, all registers
Failure detected on affected instruction – no wasted cycles



Asynchronous continuous compare



Compare retired instruction, all registers
Failure detected on affected instruction – no wasted cycles
Async events such as interrupts and debug



One test, many comparisons

```
# Info (IDV) ImperasDV VERIFICATION REPORT
# Info (IDV)    Instruction retires    : 114,943
# Info (IDV)    Traps                  : 19
# Info (IDV)    Interrupt events       : 214
# Info (IDV)    Ending cycle count     : 220,956
# Info (IDV)                                Sets / Compares
# Info (IDV)      PC                   :  114,962 / 114,943
# Info (IDV)      Instruction          :  114,962 / 114,943
# Info (IDV)      GPR                  :   54,811 / 54,811
# Info (IDV)      CSR                  :  459,687 / 231,087
# Info (IDV)      FPR                  :          0 / 0  (disabled)
# Info (IDV)      VR                   :          0 / 0  (disabled)
# Info (IDV)
# Info (IDV)    Total compares         : 515,784
# Info (IDV)    Mismatches             : 0
```

Test sources

- Architectural compliance suite
- Synopsys Imperas riscvISATESTS directed test suite
- Compiler torture suites
- Random instruction stream
  - RISC-V DV – Open source, ChipsAlliance
  - Force RISC-V – Open source, OpenHW group
  - Valtrix Sting – Commercial
  - Breker Trek-SoC – Commercial
  - RISCV-CTG – Incore, Open source

2024
DESIGN AND VERIFICATION™
DVCON
CONFERENCE AND EXHIBITION
UNITED STATES
SAN JOSE, CA, USA
MARCH 4-7, 2024

Functional Coverage

accellera
SYSTEMS INITIATIVE

riscvISACOV (Synopsys)

- Autogenerated, covers all standard extensions
- RV32I available from GitHub, others available under commercial license
- All documentation (list of cover points) is public
- Can generate coverage for custom instructions and CSRs

ISA Extension: RV32I
Specification: I Base Integer Instruction Set
Version: 2.1 XLEN: 32
Instructions: 37
Covergroups: 37
Coverpoints total: 492
Coverpoints Compliance Basic: 204
Coverpoints Compliance Extended: 176
Coverpoints DV Un-privileged Basic: 112

Extension	Subset	Instruction	Covergroup	Coverpoint	Coverpoint Description	Coverpoint Level
RV32I		add	add_cg	cp_asm_count	Number of times instruction is executed	Compliance Basic
				cp_rd	RD (GPR) register assignment	Compliance Basic

RISC-V DV

- Part of RISC-V DV test generator
- Parses trace log
- Parses all instructions, not just those checked by the DV testing
  - Leaves potential gaps in verification

```
info: PASSED test: riscv_arithmetic_basic_test_0 instructions:11463 matches:9635
info: PASSED test: riscv_ebreak_debug_mode_test_1 instructions:29982 matches:11828
info: PASSED test: riscv_mmu_stress_test_0 instructions:17896 matches:8896 means
EXECUTED
```

Incore RISC-V ISAC

- Extracts coverage from trace file and matches to coverage points
- Feeds into RISCV-CTG for test generation
- Circular dependency – tests will always hit 100% coverage (coverage is manually created and tests generated to hit that coverage)

2024
DESIGN AND VERIFICATION™
DVCON
CONFERENCE AND EXHIBITION
UNITED STATES
SAN JOSE, CA, USA
MARCH 4-7, 2024

Microarchitecture and benchmarks

accellera
SYSTEMS INITIATIVE

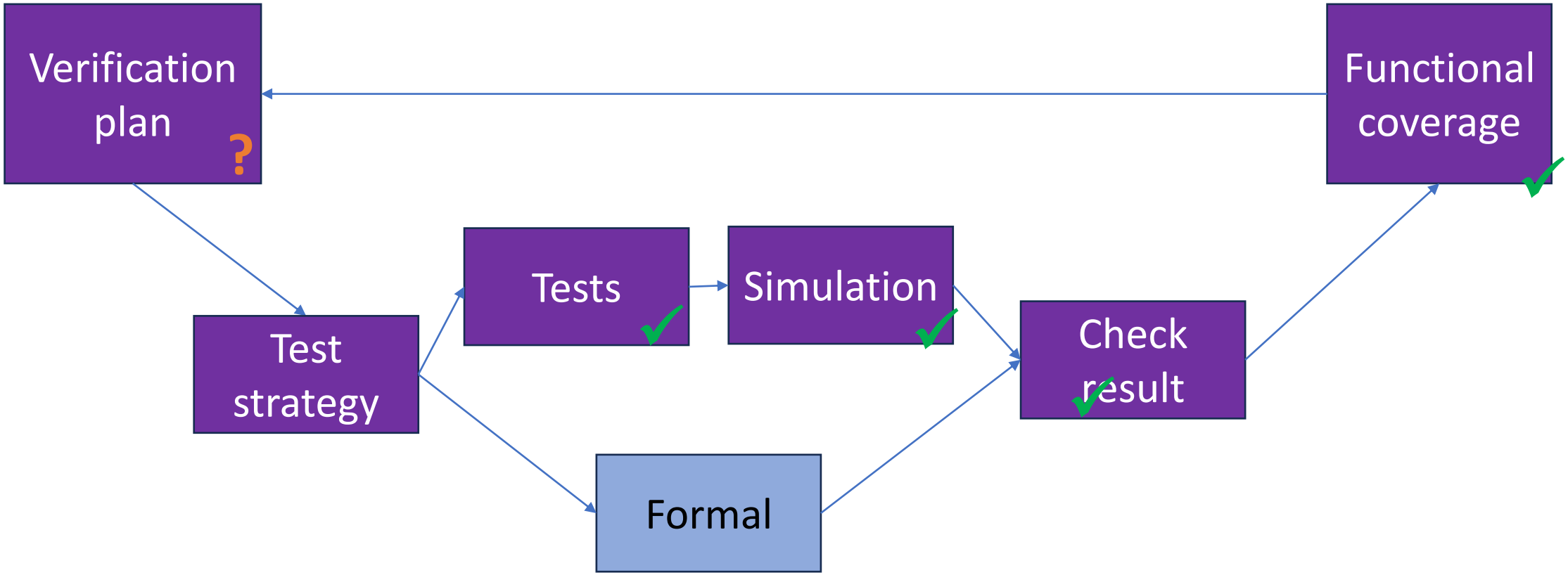
Microarchitecture and benchmarks

- Example benchmarks: Dhrystone, Embench, Coremark, Spec, Geekbench, Antutu
- Is the real application
  - Compute or data bound?
  - Floating point?
  - SIMD or vector?
  - Using special data types such as int8 or bfloat16?
- How long does it take to run? Is it feasible to run in RTL simulation?

Future work

- Improved coverage
  - Interrupts, including CSRs and which instructions are interrupted
  - Complex scenarios for MMU etc
- Integration with formal
- Compliance and certification

Conclusions



Questions?

How to combine simulation and formal for best efficiency?