PySpike: Python Bindings of RISC-V ISA Simulator

LIU Yu[1, 2] TAN Min-Qiang[1] YU Zhi-Hong[1, ✉]

[1]Wu-Xi EsionTech Inc.

[2]HuiMt Labs

RISC-V Summit China, 2024

Who we are, and what we do

EsionTech Inc.

- subsidiary of CETC's Research Institute 58, founded in 2013;
- headquarter in Wuxi, R&D centers in Beijing, Shanghai, Wuhan, . . . ;
- vendor of all-programmable and heterogeneous computing chips;
- new to RISC-V ecosystem, since early 2023;

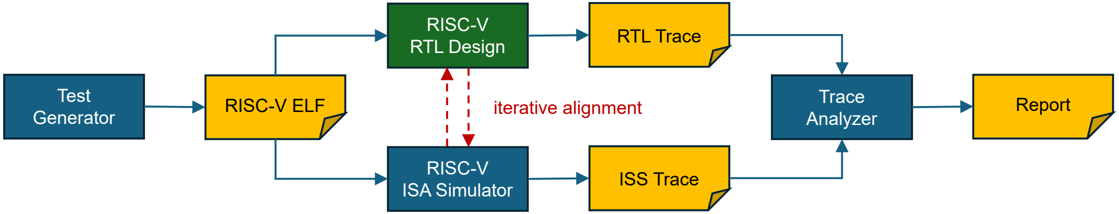
HuiMt Labs

- affiliated with EsionTech's Beijing R&D Center;
- small group of open-source software enthusiasts;
- veterans in HW / SW co-simulation and co-verification tools;
- contributors to RISC-V tools, i.e. spike, riscv-dv, . . . ;

What Spike is all about

De facto standard RISC-V ISA simulator, aka. Spike

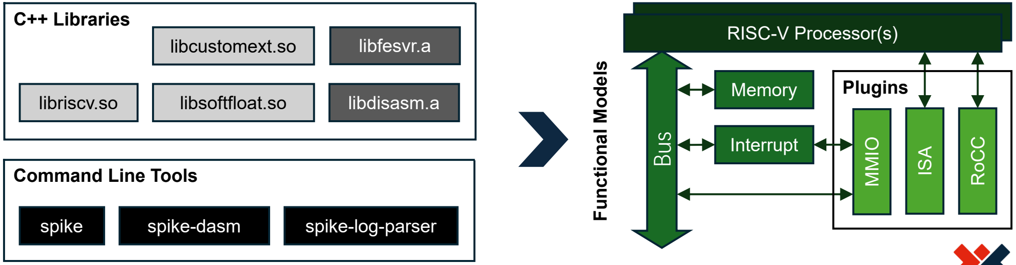
- reference model for differential testing in constrained-random verification;
  - C++ code base (40k+ lines), 14+ years of history, and state-of-the-art ISA support;
  - command line tools (spike, xspike, spike-dasm, . . .) and C++ libraries (libriscv, . . .);
  - plugin system based on dynamic loading (dlopen) of shared objects / libraries;



What Spike is all about

***De facto* standard RISC-V ISA simulator, aka. Spike**

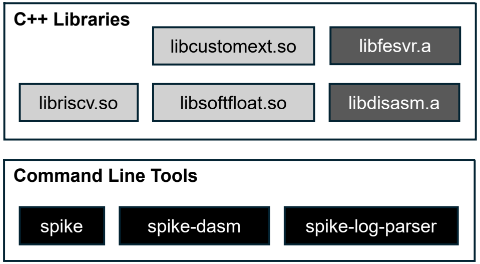
- reference model for differential testing in constrained-random verification;
- C++ code base (40k+ lines), 14+ years of history, and state-of-the-art ISA support;
- command line tools (spike, xspike, spike-dasm, . . .) and C++ libraries (libriscv, . . .);
- plugin system based on dynamic loading (dlopen) of shared objects / libraries;

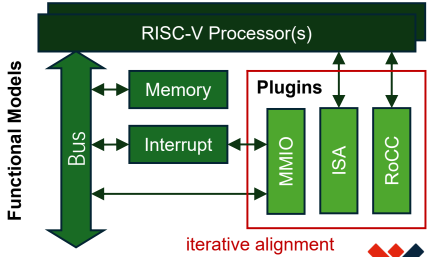


What Spike is all about

De facto **standard RISC-V ISA simulator, aka. Spike**

- reference model for differential testing in constrained-random verification;
- C++ code base (40k+ lines), 14+ years of history, and state-of-the-art ISA support;
- command line tools (spike, xspike, spike-dasm, . . .) and C++ libraries (libriscv, . . .);
- plugin system based on dynamic loading (dlopen) of shared objects / libraries;

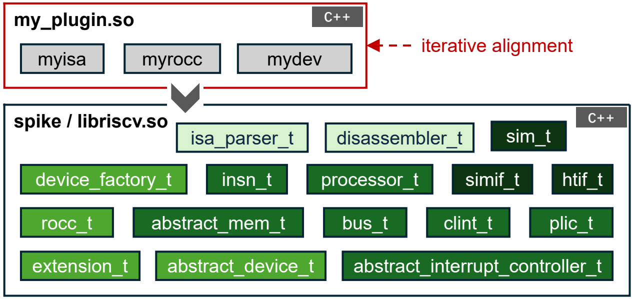




Why Spike needs Python bindings

Emerging Python-based verification calls for the agility of . . .

- fast-prototyping custom ISA extensions, accelerators, peripherals, . . . ;
- fine-granular controlling Spike instances for interoperation with testbenches;
- reusing small goodies like parsing ISA-string, disassembling opcode, . . . ;

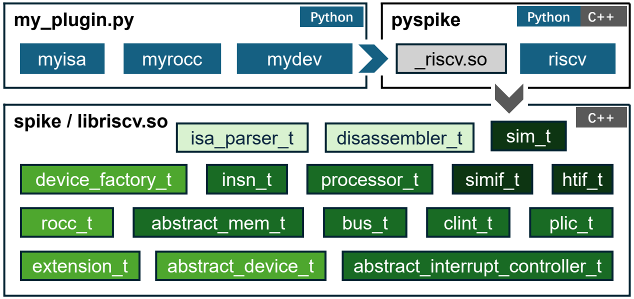


```
$ spike \
  --isa=rv64gc_xmyisa \
  --priv=msu \
  --extlib=my_plugin.so \
  --extension=myrocc \
  --device=mydev,0x20000000 \
  program.elf
```

Why Spike needs Python bindings

Emerging Python-based verification calls for the agility of . . .

- fast-prototyping custom ISA extensions, accelerators, peripherals, . . . ;
- fine-granular controlling Spike instances for interoperation with testbenches;
- reusing small goodies like parsing ISA-string, disassembling opcode, . . . ;



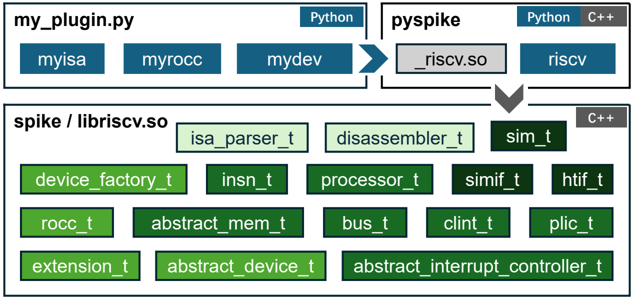
```
$ python -q
>>> from riscv.sim import *
>>> help(sim_t)
Help on class sim_t in module
↪  riscv._riscv.sim:

class
↪  sim_t(riscv._riscv.htif.htif_t,
↪  riscv._riscv.simif.simif_t)
...
```

Why Spike needs Python bindings

Emerging Python-based verification calls for the agility of . . .

- ○ fast-prototyping custom ISA extensions, accelerators, peripherals, . . . ;
- ● fine-granular controlling Spike instances for interoperation with testbenches;
- ● reusing small goodies like parsing ISA-string, disassembling opcode, . . . ;

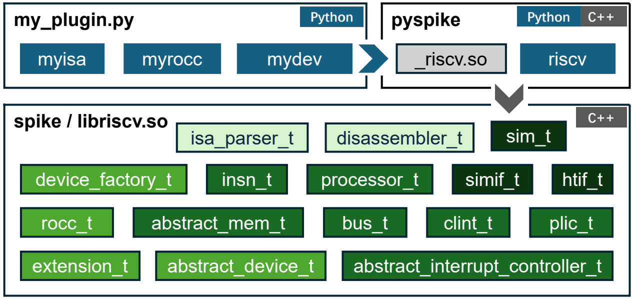


```
$ PYSPIKE_LIBS=my_plugin.py \
  spike \
  --isa=rv64gc_xmyisa \
  --priv=msu \
  --extlib=libpython3.8.so \
  --extlib=_riscv.so \
  --extension=myrocc \
  --device=mydev,0x20000000 \
  program.elf
```

Why Spike needs Python bindings

Emerging Python-based verification calls for the agility of . . .

- ○ fast-prototyping custom ISA extensions, accelerators, peripherals, . . . ;
- ● fine-granular controlling Spike instances for interoperation with testbenches;
- ● reusing small goodies like parsing ISA-string, disassembling opcode, . . . ;



```
$ pyspike \
  --isa=rv64gc_xmyisa \
  --priv=msu \
  --extlib=my_plugin.py \
  --extension=myrocc \
  --device=mydev,0x20000000 \
  program.elf
```

What PySpike has to offer, exactly

Python in Spike (PIS)

- extend vanilla Spike with ISA / RoCC / MMIO models written in Python;
- leverage Python testing frameworks, such as pytest, to improve testability of Spike and pluggable extensions;

Spike in Python (SIP)

- instantiate Spike and manipulate its internal gadgets as Python objects;
- reuse small goodies from Spike in Python scripts, including ISA-string parser, opcode disassembler, . . . ;

```
from typing import List
from riscv import insn
from riscv.disasm import *
from riscv.processor import *

@insn.register("myisa")
class MyISA(insn.ISA):
  def __init__(self): ...
  def get_instructions(self) ->
   ↪  List[insn_desc_t]: ...
  def get_disasms(self) ->
   ↪  List[disasm_insn_t]: ...
  def reset(self) -> None: ...
```

```
$ pyspike --isa=rv32gc_xmyisa
↪  --extlib=my_isa.py program.elf
```

中微亿芯 ESIONTECH

What PySpike has to offer, exactly

Python in Spike (PIS)

- extend vanilla Spike with ISA / RoCC / MMIO models written in Python;
- leverage Python testing frameworks, such as pytest, to improve testability of Spike and pluggable extensions;

Spike in Python (SIP)

- instantiate Spike and manipulate its internal gadgets as Python objects;
- reuse small goodies from Spike in Python scripts, including ISA-string parser, opcode disassembler, . . . ;

```
from typing import Optional
from riscv import mmio
from riscv.sim import sim_t

@mmio.register("mydev")
class MyDEV(mmio.MMIO):
  def __init__(self, sim: sim_t,
  ↪  args: Optional[str]): ...
  def load(self, addr: int, size:
  ↪  int) -> bytes: ...
  def store(self, addr: int, data:
  ↪  bytes) -> None: ...
```

```
$ pyspike --isa=rv32gc
↪  --extlib=mydev.py
↪  --device=mydev,0x20000000
↪  program.elf
```

What PySpike has to offer, exactly

Python in Spike (PIS)

- ○ extend vanilla Spike with ISA / RoCC / MMIO models written in Python;
- ○ leverage Python testing frameworks, such as pytest, to improve testability of Spike and pluggable extensions;

Spike in Python (SIP)

- ● instantiate Spike and manipulate its internal gadgets as Python objects;
- ○ reuse small goodies from Spike in Python scripts, including ISA-string parser, opcode disassembler, . . . ;

```
from riscv.cfg import *
from riscv.sim import sim_t


s = sim_t(
  cfg=cfg_t(
    isa="rv32gc",
    mem_layout=[
      mem_cfg_t(0x90000000, 0x40000)
    ]),
  halted=False,
  plugin_device_factories=[
    ("mydev", ("0x20000000", ))
  ],
  args="program.elf")


...
```

What PySpike has to offer, exactly

Python in Spike (PIS)

- extend vanilla Spike with ISA / RoCC / MMIO models written in Python;
- leverage Python testing frameworks, such as pytest, to improve testability of Spike and pluggable extensions;

Spike in Python (SIP)

- instantiate Spike and manipulate its internal gadgets as Python objects;
- reuse small goodies from Spike in Python scripts, including ISA-string parser, opcode disassembler, . . . ;

```
from riscv.decode import insn_t
from riscv.isa_parser import *
from riscv.disasm import *

def test_misc():
  p = isa_parser_t("rv64gc_xmyisa",
  ↪  "msu")
  assert ord('F') in p
  assert "myisa" in p

  d = disassembler_t(p)
  x = insn_t(b"\x13\x86\x82\x02")
  assert d.disassemble(x) == "addi
  ↪  a2, t0, 40"
  assert x.i_imm == 40
```

```
$ pytest -v
```

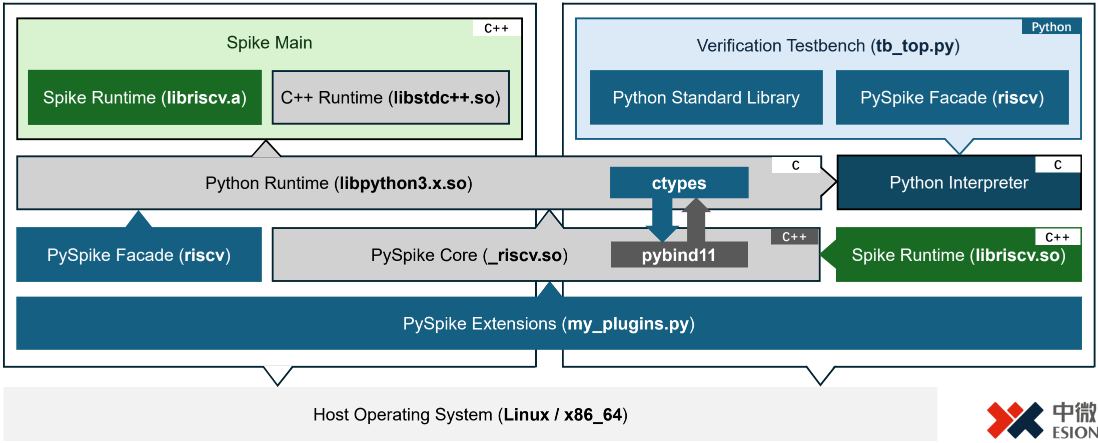
How PySpike enables dual bindings

Python in Spike (PIS)

- plug Python code into vanilla Spike;

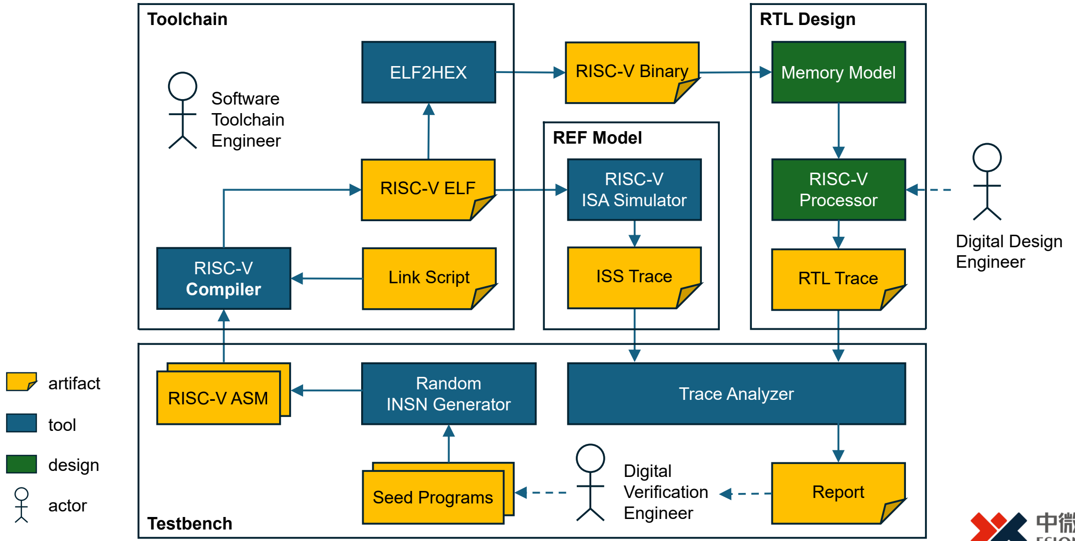
Spike in Python (SIP)

- access Spike internals from Python;

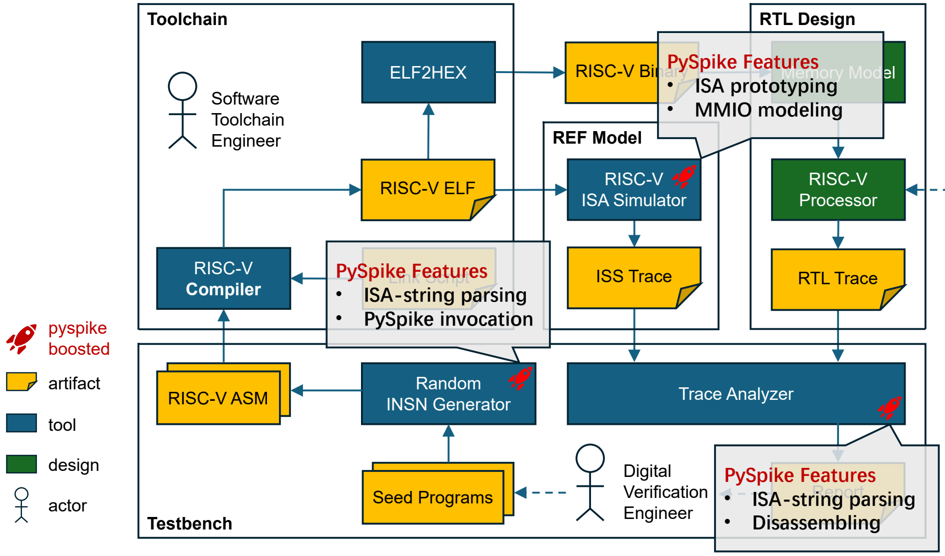


C++
Spike Main
Spike Runtime (libriscv.a)
C++ Runtime (libstdc++.so)
Python
Verification Testbench (tb_top.py)
Python Standard Library
PySpike Facade (riscv)
C
Python Runtime (libpython3.x.so)
ctypes
C
Python Interpreter
PySpike Facade (riscv)
PySpike Core (_riscv.so)
pybind11
C++
C++
Spike Runtime (libriscv.so)
PySpike Extensions (my_plugins.py)
Host Operating System (Linux / x86_64)

Where PySpike boosts HW verification



Where PySpike boosts HW verification



We are releasing PySpike to the public!

Our belief in openness and sharing

- PySpike opens up Spike's C++ internals for interoperation with Python; we believe it can help boost the agility of Python-based hardware verification;
- As a language binding, PySpike needs the community's help to keep up with Spike's changes to its evolving feature set and *unstable* C++ API's.

When and where to obtain PySpike

- pending approval from EsionTech, will be announced on HuiMt Labs' website;
- for those interested in early access, please contact us on GitHub (huimtlab).

中微亿芯 ESIONTECH

References I

Andrew Waterman, et al.
Spike RISC-V ISA Simulator
https://github.com/riscv-software-src/riscv-isa-sim

Yinan Yu, et al.
Towards Developing High Performance RISC-V Processors Using Agile Methodology
IEEE/ACM International Symposium on Microarchitecture (MICRO), 2022.

William M. McKeeman.
Differential Testing for Software
Digital Technical Journal, Vol. 10(1), 1998.

李枫
基于 Python 的硬件验证
https://www.bilibili.com/video/BV1LV411X7SN/