Vyoma

Verification Makeover with RISC-V Processor Designs

Lavanya J | CEO/Founder | Vyoma

Vyoma

Intro

12+ years of industry experience (with IBM, ARM & Rambus) and academic with an MS from IIT Madras. Passionate about making design verification fun for the masses!

IITM SHAKTI RISC-V Efforts

Vyoma - VaaS Platform for RISC-V Verification

Incubated @ IITM Pravartak Technologies Foundation

DIR-V Startup

Vyoma

Agenda

RISC-V Processor Verification as a foundation to provide the next-gen verification methodology and infrastrucure

RISC-V Business & Verification

Makeover Components

SHAKTI Verification

Vyoma's UpTickPro

Vyoma

Quality

Time to Market

Customization

RISC-V Business

Success of the design companies in the growing RISC-V Processor and Subsystem business

RISC-V Business

RISC-V Verification

Vyoma

Quality

Test and Coverage Quality

Time to Market

Methodology and Infrastructure

Customization

Verification IPs

What does it mean for Verification?

Vyoma

TEST & TEST TEMPLATES

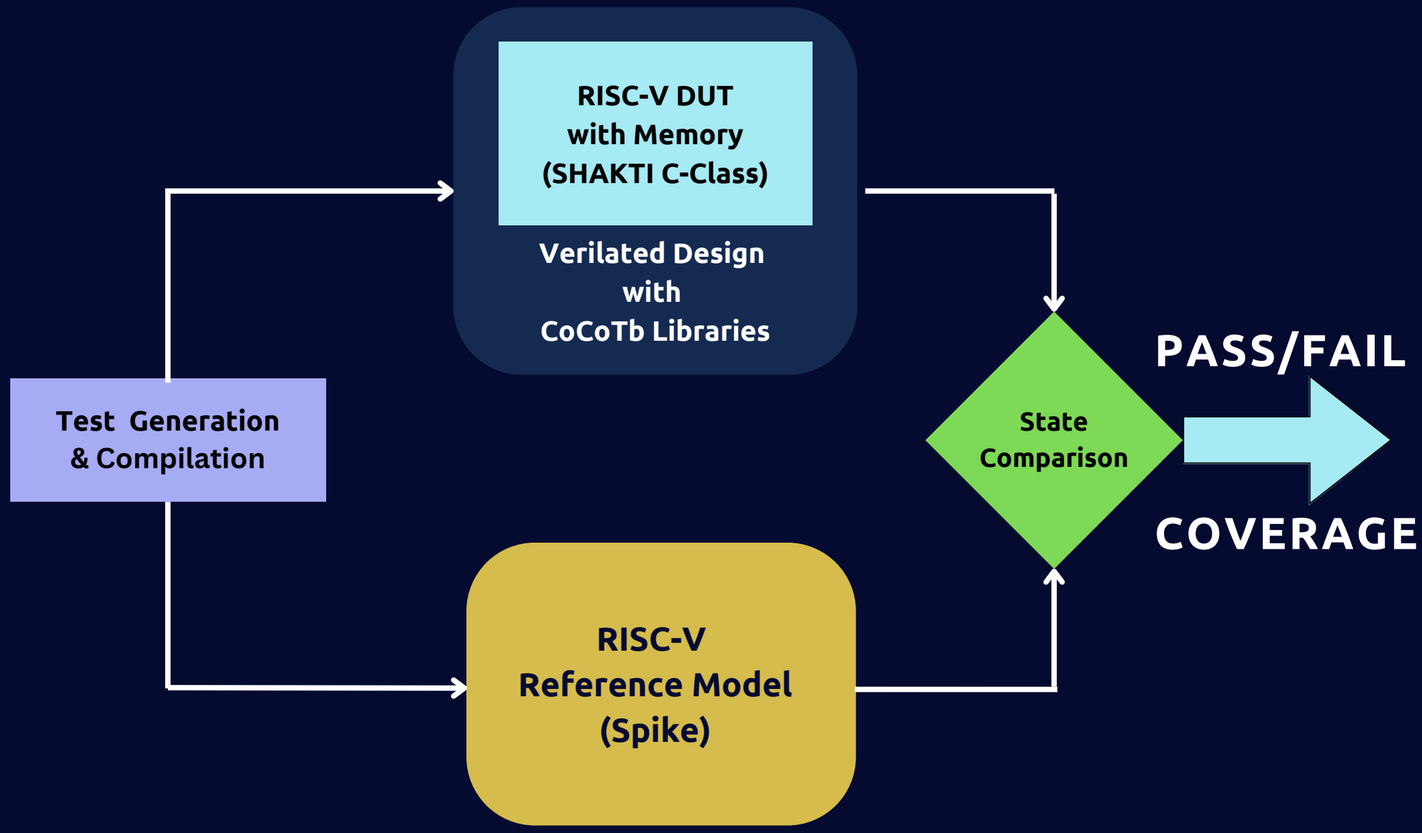
TEST BENCH

REFERENCE MODEL

SCOREBOARD

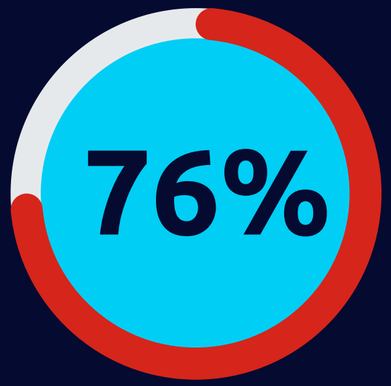
TOOLCHAIN & INFRA

COVERAGE

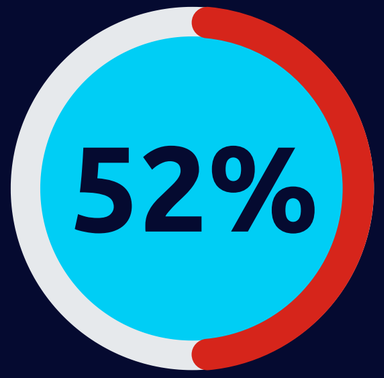


The Wakeup Call

Vyoma



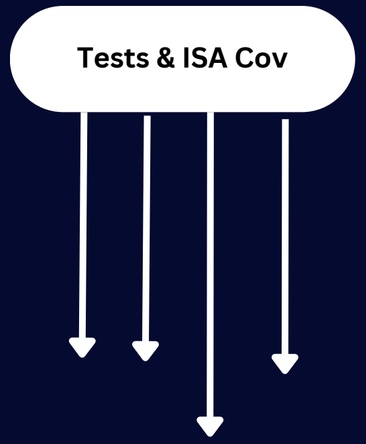
76% of today's designs **DO NOT achieve first-silicon success**



Logic and functional flaws are the leading cause of re-spins

Source: Wilson Research Group and Siemens EDA, 2022 Functional Verification Study

Makeover Tests & Coverage



Configurable Test Plugins

Re-use the Python Ecosystem

Generative AI Tests

- One generator does not filt ALL scenarios.
- Leverage open-source RISC-V generators , customize for RISC-V Designs & Parallelize it!
- Ready to deploy VIPs in the form of test templates or plugins and their quality defined using the ISA coverage metrics

Chatbotting Tests and their Coverage!

Makeover Test Bench

Vyoma

Constraint based randomization ✓

```python
class FinalRandomized(Randomized):
  def __init__(self, x):
      Randomized.__init__(self)
      self.x = x
      self.y = 0
      self.z = 0

      # define y as a random variable taking values from 0 to 9
      add_rand("y", list(range(10)))

      # define z as a random variable taking values from 0 to 4
      add_rand("z", list(range(5)))

      # hard constraint
      add_constraint(lambda x, y: x !=y)
      # multi-dimensional distribution
      add_constraint(lambda y, z: y + z)

# create randomized object instance (default values at this point)
obj_ = FinalRandomized(5)
# randomize object with additional contraint
obj_.randomize_with(lambda z : z > 3)
```

Pythonize Test Bench

Makeover Test Bench

Vyoma

Constraint based randomization ✓

```python
class InputDriver(BusDriver):
    """Drives inputs to DUT."""

    _signals = [<input_interface>] # Input interface signals

    def __init__(self, dut):
      BusDriver.__init__(self, dut, None, dut.CLK)

    ...

self.input_drv = InputDriver(dut)  # Driver instantiation

...

yield tb.input_drv.send(<input_transaction>)  # Sending the transaction

```

Test Bench Components ✓

```python
class MyScoreboard(Scoreboard):
    def compare(self, got, exp, log, **_):
        if got != exp:
            self.errors += 1
            log.error("Received transaction differs from expected output.")
            if self._imm:
                raise TestFailure("DUT differs from model.")
...


self.scoreboard = MyScoreboard(dut, fail_immediately=True)
self.scoreboard.add_interface(self.output_monitor, self.expected_output, srict_type=False)

```

```python
class OutputMonitor(BusMonitor):
    """Observes outputs of DUT."""
    _signals = [<signal list to be monitored>]

    def __init__(self, dut, tb, callback=None, event=None):
        BusMonitor.__init__(self, dut, None, dut.CLK,
                            dut.RST_N, callback=callback, event=event)
        self.name = "out"
        self.tb = tb

    @coroutine
    def _monitor_recv(self):
        clkedge = RisingEdge(self.clock)
        while True:
            yield clkedge
            self._recv(OutputTransaction(self.tb, <monitred_signals>))

        ...

# Instantiation
self.output_monitor = OutputMonitor(dut, self)
```

Pythonize Test Bench

Makeover Test Bench

Vyoma

Constraint based randomization ✓

Test Bench Components ✓

Coverage Definition ✓

```python
import cocotb
from cocotb_coverage.coverage import *

Adder_Coverage = coverage_section (
  CoverPoint("top.a", vname="a", bins = list(range(0,16))),
  CoverPoint("top.b", vname="b", bins = list(range(0,16))),
  CoverCross("top.cross_cover", items = ["top.a", "top.b"])
)

@Adder_Coverage
def adder_model(a: int, b: int) -> int:
    """ model of adder """
    return a + b
```

Pythonize Test Bench!

Makeover Tools and Infrastructure

Containerized environment for different designs

Same Infra for Development, Regression and Production release

Reliably Scalable for leveraging the cloud compute

Dockerize the Infra!

Makeover Verification

VIP Plugins
and
Generative AI tests

Efficiency

Pythonize

Productivity

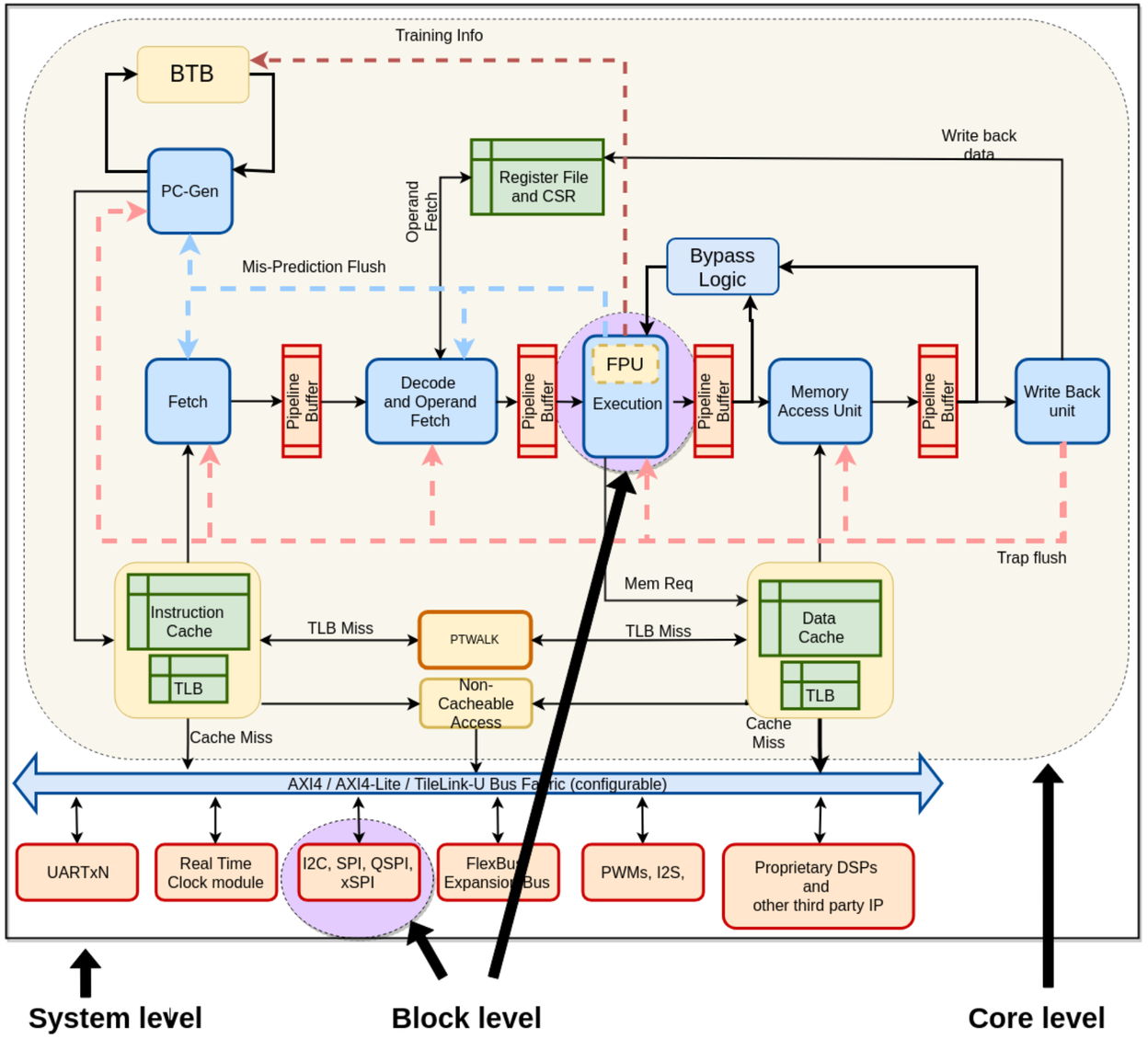
Dockerize

Scalability

Ensures High Quality Designs a Reality

SHAKTI C-Class

In-Order RISC-V



SHAKTI Design Build

Vyoma

TEST & TEST TEMPLATES

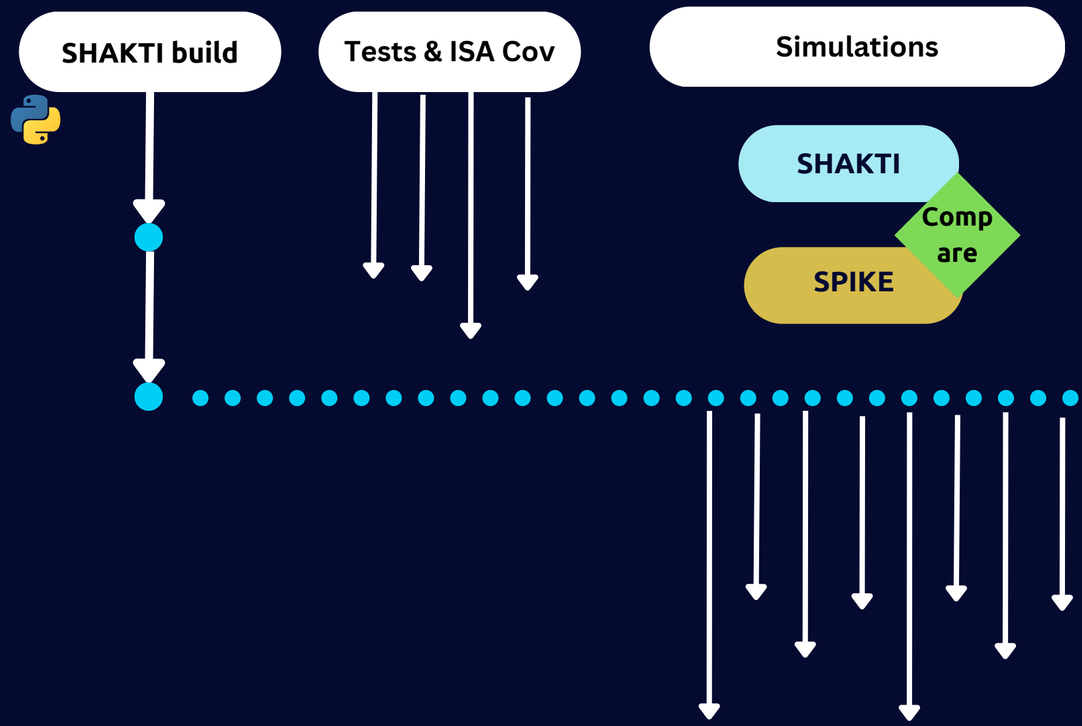
TEST BENCH

REFERENCE MODEL

SCOREBOARD

TOOLCHAIN & INFRA

COVERAGE



- RTL Built for different configurations and verification abstractions
- Configurations can be different test memory sizes, simulation and FPGA builds
- Design configurations: with or without debug, micro-arch configurations

Test Templates

Vyoma

TEST & TEST TEMPLATES

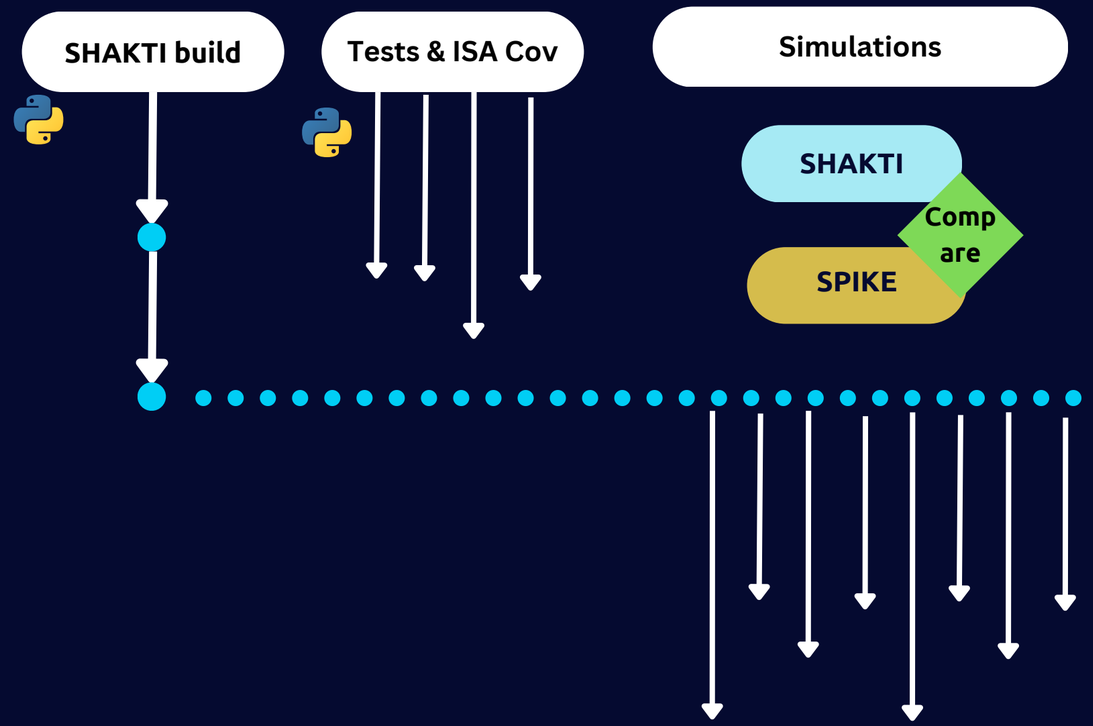
TEST BENCH

REFERENCE MODEL

SCOREBOARD

TOOLCHAIN & INFRA

COVERAGE



- One generator does not filt ALL scenarios.
- Leverage open-source RISC-V generators , customize for SHAKTI Designs & Parallelize it!
- Ready to deploy VIPs in the form of test templates and their quality defined using the ISA coverage metrics

Test Bench & Simulations

Vyoma

TEST & TEST TEMPLATES

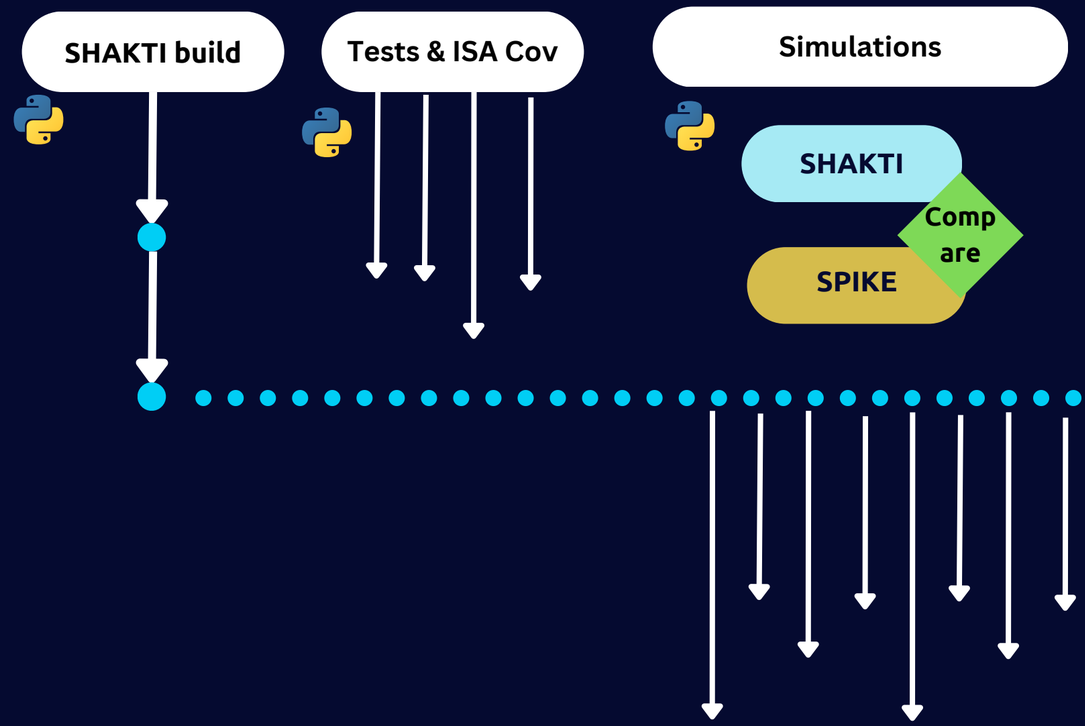
TEST BENCH

REFERENCE MODEL

SCOREBOARD

TOOLCHAIN & INFRA

COVERAGE



- CoCoTb based Python test benches following UVM methodology
- On-the-fly Processor state comparison
- Utmost care taken to re-use and make it configurable
- **Simulations -> Parallelize, Parallelize, Parallelize!**

Ground Realities

Vyoma

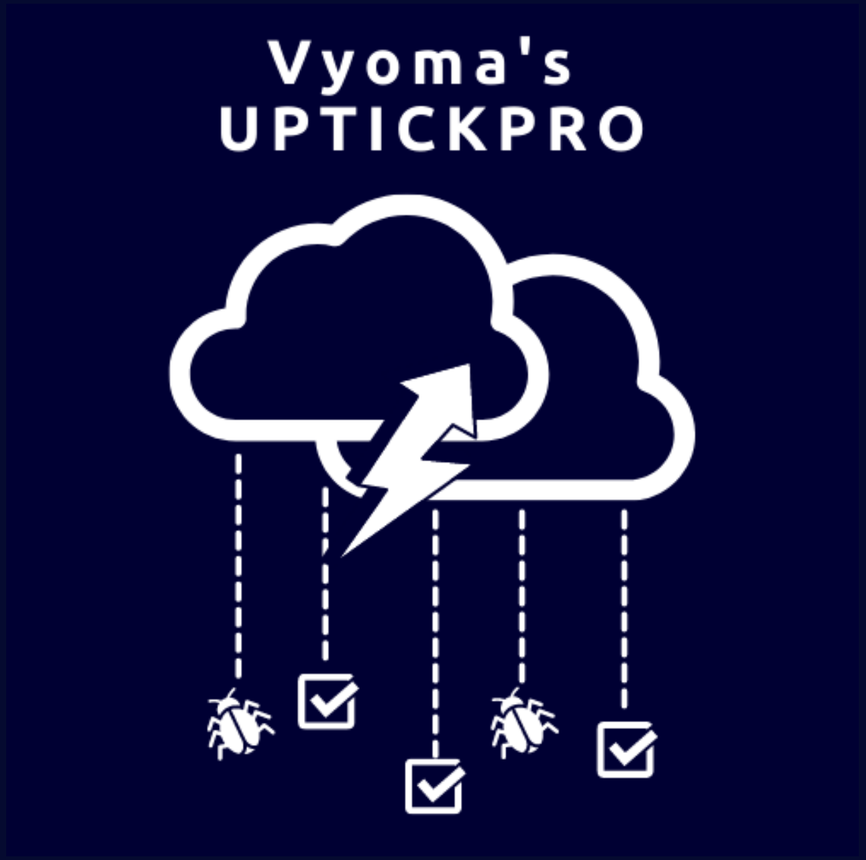
- Apple to Apple language or Performance Comparisons
- Assertions, TB debug capability in the waveforms
- ? Can handle complexity ? In-order/OOO Cores/ Multi-core (WIP)
- \+ Open source and Python Ecosystem support 
- \+ Productivity: Faster ramp-up, Rapid **Verif DevOps**

Verification of the Future!

Vyoma: Verification-as-a-Service Platform

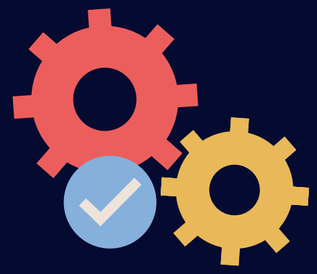
Vyoma

Ensures RISC-V Design Quality



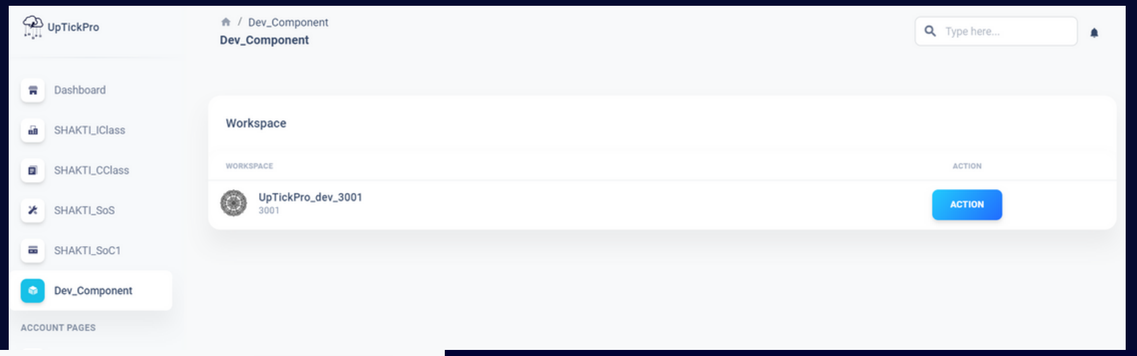
Test suites & Generators for end-to-end verification completeness

Scalable Test Infrastructure

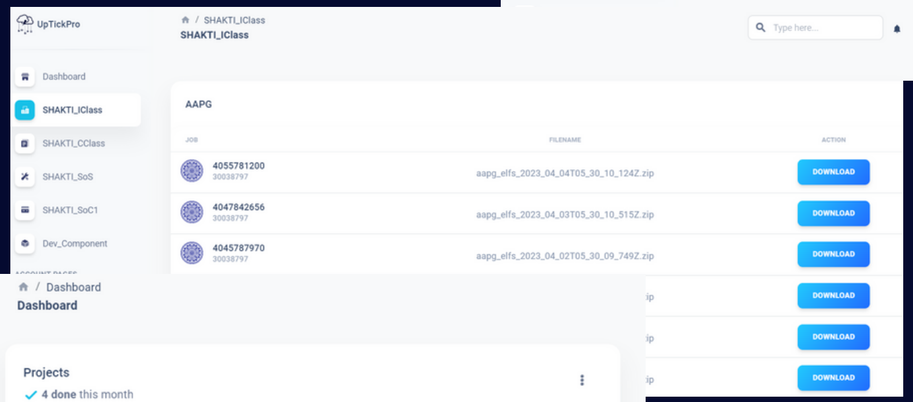


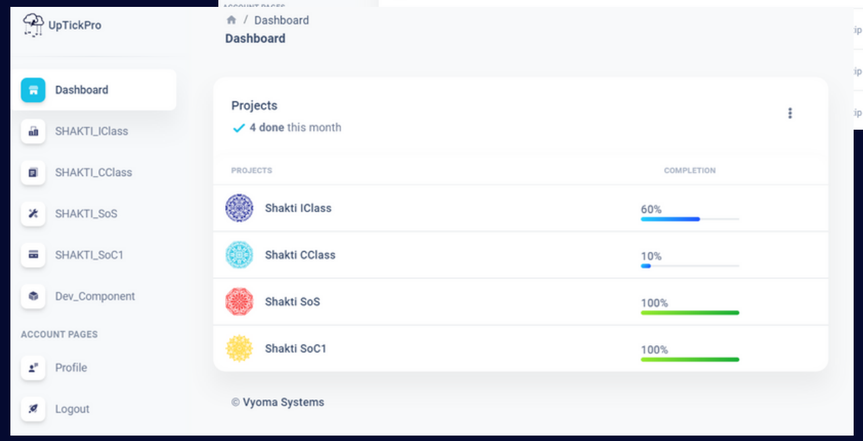
UpTickPro: Verification DevOps

Vyoma

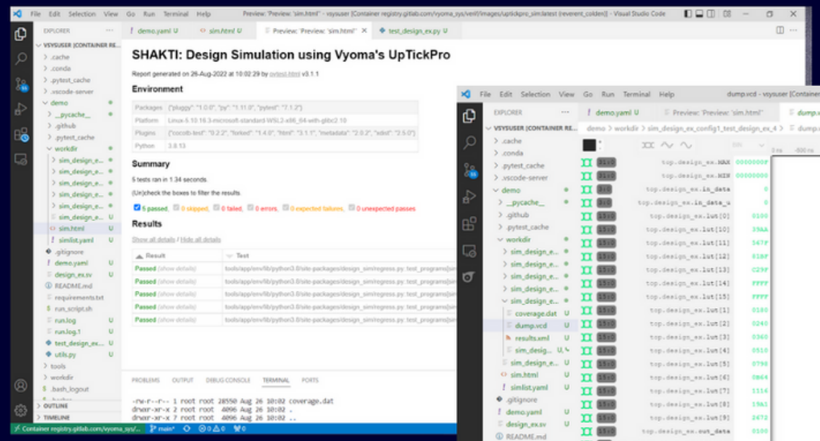


DEV ENVIRONMENT

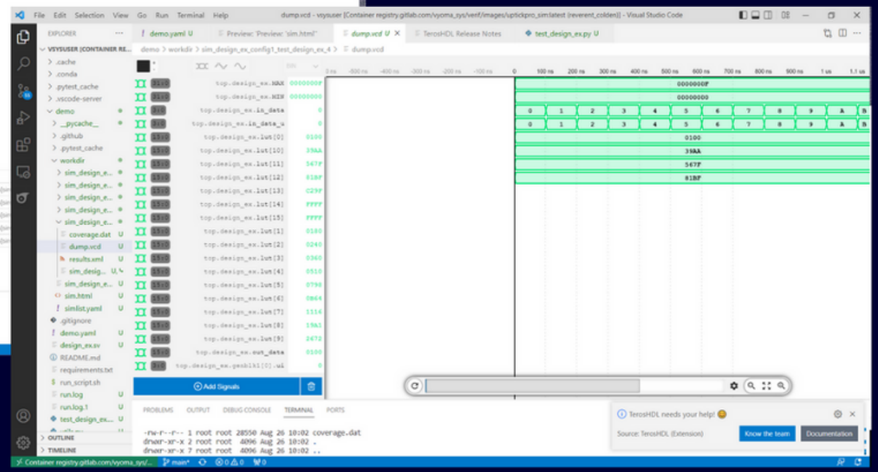




VERIF DASHBOARD

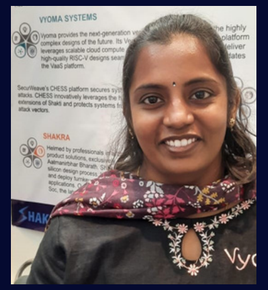


HTML REPORTS



WAVEFORMS

Team



Lavanya J

CEO/Founder

11+ years of industry experience (with IBM, ARM & Rambus) with an MS from IIT Madras. Passionate about making design verification fun for the masses!

Rizwana G

SaaS Lead Architect

7+ years of professional experience in the software field with an undergrad degree from Anna University. Owns DevSecOps for Vyoma's SaaS Implementation.

Prof. V. Kamakoti

Honorary Mentor

Director at IIT Madras. Visionary for a semiconductor Atma-Nirbhar Bharat. Pioneering India’s first RISC-V processor designs (SHAKTI)

Sundara Nagarajan

Scaleup Advisor

An industry veteran in the Computer Systems industry; career technologist, and product development leader. Currently Managing Director of Innovation Scaleup Advisors, a firm that empowers tech founders to build scalable firms.

Vyoma

Thank You

Contact Us:

lavanya@vyomasystems.com

https://vyomasystems.com/